



**This electronic thesis or dissertation has been
downloaded from Explore Bristol Research,
<http://research-information.bristol.ac.uk>**

Author:
Arabul, Ekin

Title:
**A Precise High Count-Rate Multi-Channel Coincidence Counting Instrument for
Quantum Photonics Applications**

General rights

Access to the thesis is subject to the Creative Commons Attribution - NonCommercial-No Derivatives 4.0 International Public License. A copy of this may be found at <https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>. This license sets out your rights and the restrictions that apply to your access to the thesis so it is important you read this before proceeding.

Take down policy

Some pages of this thesis may have been removed for copyright restrictions prior to having it been deposited in Explore Bristol Research. However, if you have discovered material within the thesis that you consider to be unlawful e.g. breaches of copyright (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please contact collections-metadata@bristol.ac.uk and include the following information in your message:

- Your contact details
- Bibliographic details for the item, including a URL
- An outline nature of the complaint

Your claim will be investigated and, where appropriate, the item in question will be removed from public view as soon as possible.



**A Precise High Count-Rate Multi-Channel
Coincidence Counting Instrument for Quantum
Photonics Applications**

By: Ekin Arabul

Supervised by: Dr. Naim Dahnoun & Prof. John Rarity

*A dissertation submitted to the University of Bristol in accordance with
the requirements of the degree of Doctor of Philosophy in the Faculty of
Engineering.*

Department of Electrical & Electronic Engineering

University of Bristol

December 2019

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: DATE:

Abstract

Coincidence counters play the role of gating the correlated events from the background noise in almost every quantum photonics setup. To provide flexibility and precision in experiments, commonly Time-to-Digital Converters (TDCs) are utilised for implementing coincidence counters. TDCs are instruments used for converting time events into digital numbers, and typically in quantum photonics experiments, they are used for the time-stamping of photons' arrival times. With recent developments in quantum photonics, precise multi-channel and high count-rate coincidence counting tools have become desirable due to the increase in the complexity of quantum photonics experiments. However, timing analyser instruments are struggling to meet with the performance requirements of these experiments and becoming limiting factors.

In this research, Field Programming Gate Array (FPGA) based methods have been researched to integrate a precise multi-channel TDC with a coincidence counting instrument into the same FPGA fabric. To achieve this, a 512-bin carry chain based tapped delay line TDC has been developed in a Spartan6 LX150 FPGA. The developed TDC scheme (Dual Data-Rate Registration TDC) was unique in a way that it uses both clock edges to register the trigger and applying an averaging technique in combination with the code density calibration to improve the linearity. The advantages of this method were without changing the number of delay lines used or introducing an additional dead-time, better linearity was achieved. By using this method 8.9 ps Single Shot Precision (SSP) (with a bin width of 7.7 ps), 12.6 ps standard deviation, Full-Width-Half-Maximum (FWHM) of 29.6 ps, the max DNL error of 2.9 LSB and max INL of 8.8 LSB were achieved.

This TDC scheme was used to implement a multi-channel labelling coincidence counting method. This method integrates the TDC and coincidence counter implementations into the same chip while using a multi-stop LiDAR approach to achieve optimum real-time operation. This system provided 8-channels with adjustable digital delays and window sizes for each channel. The coincidence counting operation achieved 40 million counts per second (MCPS), and for 8 operational channel, the total of 320 MCPS was achieved. This result was the best-achieved count-rate for an 8 or more channel coincidence counting system with a sub-10 ps precision. Also, the smallest window size which could precisely capture all coincidences were observed as 107 ps, while the settable smallest window was 7.7 ps. Also, the system was successfully tested in quantum photonics applications such as the rev-HOM dip measurement and the pseudo-photon-number resolving detection of a coherent state of light.

The future work for this research involves improving the precision by adding multiple phases to the TDC registration scheme, improving the coincidence counting implementation for an additional number of channels and improving the count-rate of the system by developing a dual data-rate TDC.

Table of Contents

	Page
Dedication	xi
Acknowledgements	xii
List of Tables	xv
List of Figures	xvii
List of Acroynms	xxiii
Nomenclature	xxvii
Publications	xxix
1 Introduction	1
1.1 Background	3
1.2 Aims & Constraints	4
1.2.1 Timing Precision	5
1.2.2 Maintaining Real-Time Operation	6
1.2.3 Data Transfer Throughput	7
1.3 Thesis Outline	8
1.4 Research Contributions	10
2 Research Review	13
2.1 Introduction	13
2.2 Time-of-Flight Measurements	13
2.2.1 LiDAR Overview	14
2.2.1.1 Modulation LiDAR	14
2.2.1.2 Pulse LiDAR	20
2.2.1.3 Single-Stop LiDAR	21
2.2.1.4 Multi-Stop LiDAR Technique	22
2.2.2 LiDAR Summary	24

TABLE OF CONTENTS

2.3	Quantum Information	25
2.3.1	Quantum Bits	26
2.3.2	Quantum Interference	27
2.3.3	Hong-ou-Mandel Effect	28
2.3.4	Quantum Communication	30
2.4	Technologies Available for Timing Measurement Tool	33
2.4.1	Field Programmable Gate Arrays (FPGAs)	34
2.4.2	Application-Specific Integrated Circuits (ASICs)	35
2.4.3	Analogue Circuits	36
2.5	Time-to-Digital Converters (TDCs)	38
2.5.1	Coarse Counter TDC	39
2.5.2	Hybrid Interpolation	41
2.5.3	Fine Time TDC Schemes	42
2.5.3.1	Time-to-Analogue Converters	42
2.5.3.2	Time Expansion Method	44
2.5.3.3	Tapped Delay Lines	46
2.5.3.4	Vernier Delay Lines	49
2.5.3.5	Pulse Shrinking TDC	53
2.5.3.6	Local Passive Interpolation	54
2.5.3.7	Stochastic TDC	55
2.5.3.8	Successive Approximation TDC	57
2.5.3.9	Gated Ring Oscillators	58
2.5.3.10	Algorithmic TDC	59
2.5.3.11	SERDES TDC	61
2.5.3.12	Wave Union Launchers	61
2.5.3.13	Digital Signal Processor Based TDC	64
2.5.4	Measurement Errors and Precision of a TDC	65
2.5.4.1	Precision Parameters of a TDC	65
2.5.4.2	Offset and Gain Errors	66
2.5.4.3	Linearity of a TDC	67
2.5.4.4	Integral Non-Linearity (INL) and Differential Non-Linearity (DNL) of a TDC	69
2.5.4.5	Quantisation Errors	70
2.5.4.6	Metastability of a TDC	72
2.5.5	TDC Calibration & Linearisation Techniques	73
2.5.5.1	Direct Calibration	74
2.5.5.2	Code Density Testing Calibration	74
2.5.5.3	Equivalent Coding Lines	76
2.5.5.4	Averaging TDC Technique	77
2.5.5.5	Sliding Scale Technique	79

2.5.6	Time-to-Digital Converter Summary	80
2.6	Coincidence Counters	82
2.6.1	Analogue Coincidence Detection and Counting Circuits	82
2.6.2	Pulse Height Analysing Coincidence Counter	84
2.6.3	Height-To-Width Converter Coincidence Counter	85
2.6.4	Combinatorial Logic Based Coincidence Counters	87
2.6.5	TDC based Coincidence Counters	90
2.6.5.1	Rolling Window Method	92
2.6.5.2	Forward And Backward Looking Tag Differences Meth- ods	93
2.6.5.3	Multi-Channel Labelling Coincidence Counter	95
2.6.6	Superconducting Single Flux Quantum Coincidence Circuits . .	97
2.7	Summary	98
3	Legacy Coincidence Counters	101
3.1	Introduction	101
3.2	Software based Real-Time Coincidence Counting System	102
3.2.1	Software Implementation	103
3.2.2	Summary	105
3.3	Backward Looking Tag Difference FPGA based Real-Time Coincidence Counter	105
3.3.1	Introduction	105
3.3.2	Time-to-Digital Converter (TDC) Implementation	106
3.3.3	Coincidence Counter Implementation	107
3.3.4	Summary	107
3.4	FPGA based Forward Looking Tag Difference Coincidence Counter using Tag Serialisation	108
3.4.1	Introduction	108
3.4.2	System Overview	109
3.4.3	Time-to-Digital Converter (TDC) Implementation	109
3.4.4	Coincidence Counter	110
3.4.5	Summary	111
3.5	Conclusion	111
4	Time-to-Digital Converter Implementation	113
4.1	Introduction	113
4.2	Time-to-Digital Converter (TDC) Implementation	113
4.3	TDC calibration and Linearity Improvements	115
4.3.1	Double Data Rate Registration Linearisation	115
4.3.1.1	Tag Selector Module	117

TABLE OF CONTENTS

4.3.2	Self Calibrating TDC	118
4.3.2.1	Code Density Testing Calibration	119
4.3.3	Trigger Generation	123
4.4	Summary	124
5	A Precise High Count-Rate Multi-Channel Coincidence Counting System	127
5.1	Introduction	127
5.2	System Overview	127
5.3	Coincidence Counter Implementation	128
5.3.1	Multi-Tag Correlator	130
5.3.2	Independent Coincidence Detectors	131
5.3.3	Independent Coincidence Counting Blocks	132
5.3.4	Sub-fold Coincidence Identification Algorithm and Channel Labelling	135
5.4	Coincidence Counter Hardware	137
5.5	Coincidence Counter Software Interface	139
5.6	Summary	141
6	Functionality, Linearity and Precision of the Time-to-Digital Converter	143
6.1	Introduction	143
6.2	Timing Jitter Measurements	144
6.3	Time-to-Digital Converter (TDC) Linearity	146
6.3.1	Bin Characterisation of the TDC	147
6.3.2	Differential Non-Linearity (DNL) of the TDC	149
6.3.3	Integral Non-Linearity of the TDC	151
6.4	Transfer Function of the TDC	152
6.5	Multi-Stop TDC	155
6.6	Summary	156
7	Coincidence Counting Benchmarks and Applications	159
7.1	Introduction	159
7.2	Coincidence Counting Benchmarks	160
7.2.1	Coincidence Rate As A Function of the Delay	160
7.2.2	Linearity of the Coincidence Counter	161
7.3	Quantum Photonics Applications	163
7.3.1	Comparison with Picoharp300	163
7.3.2	Rev-HOM Dip Measurement in Two Single-Photon Interference	163
7.3.3	Pseudo-photon-number Resolving Detection of a Coherent State of Light	166

TABLE OF CONTENTS

7.4	Summary	168
8	Conclusions	171
8.1	Conclusion	171
8.2	Summary	173
8.3	Future Works	176
8.3.1	Improving Precision and Linearity of the TDC	176
8.3.2	Dual Data Rate TDC	177
8.3.3	Improving Coincidence Counter Channel Number	179
	Bibliography	181
	Appendix A : Results from Legacy Coincidence Counters	203
	Introduction	203
	Real-Time Coincidence Counting Software	203
	Coincidence Counting Benchmarks	203
	Backward Looking Tag Difference Real-Time FPGA based Coincidence Counter	204
	Coincidence Counting Benchmarks	204
	Real-Time Forward Looking Tag Difference FPGA using Tag Serialisation . .	206
	Coincidence Counting Benchmarks	206
	Appendix B : Experiments Results From Pseudo-photon-number Re-	
	solving Detection of a Coherent State of Light	209

Dedication

To my family & friends

Acknowledgements

First of all, I would like to thank Dr Naim Dahnoun and Prof John Rarity for all the support and guidance they provided me during my PhD, and giving me a chance to specialise my skills and become a better engineer. Secondly, I would like to thank, Dr Xiao Ai, for teaching me the fundamental technical knowledge I needed to conduct this research. For always being patient, morally and financially supportive, I would like to thank my mum, dad and brother, without their support, this journey would not be possible. Moreover, I would like to thank Emek Electrical Industry Inc. for funding all my undergraduate and postgraduate studies for a decade.

I would also like to thank the people in the university who supported me academically and technically during this research and helped me to overcome the obstacles I faced. I would like to thank Scott Tancock for generously helping me with all theoretical and technical problems I faced during the PhD. For giving me a chance to test my design in Quantum Photonics scenarios to see its actual potential, I would like to thank Dr. Stefano Paesani. I would like to thank Dr Stefan Frick for helping me with testing my design, and showing me its technical limitations. Also, I would like to thank Joseph Lennon for providing a breakout board to make the multi-channel timing instrumentation research possible.

Finally, I would like to thank Ufuk Erol for always being a close friend and supporting me throughout my PhD and making it manageable and enjoyable.

List of Tables

TABLE	Page
2.1 <i>Comparison of TDCs in the reviewed literature. N/A = Not Available. *</i> <i>CMOS ASICs in nm, FPGAs by series number [1].</i>	81
5.1 <i>Spartan 6 LX150 Features [2]</i>	138
8.1 <i>Performance parameters of the proposed coincidence counting system and existing TDC based coincidence counters</i>	172
1 <i>Measurement results from Pseudo-photon-number Resolving Detection of a Coherent State of Light (Attenuation vs N-fold Coincidence Rate/s)</i>	210

List of Figures

FIGURE	Page
1.1 <i>The example usage of a coincidence measurement in a quantum application [3]</i>	2
1.2 <i>Illustration of timing accuracy and precision</i>	6
1.3 <i>Places where the real-time operation can be interrupted in a coincidence counting system</i>	7
1.4 <i>The Illustration of the data transfer throughput, a : The Data Serialisation between data generation and processing, b: The graph of transferred and generated data</i>	8
2.1 <i>Demonstration of the phase difference between waves and the relationship between the phase and charge stored in capacitors [4]</i>	16
2.2 <i>Illustration of a Sawtooth Frequency Modulated Carrier Wave LiDAR</i>	17
2.3 <i>Illustration of a Triangle Wave Frequency Modulated Carrier Wave LiDAR</i>	18
2.4 <i>Illustration of an AMCW LiDAR</i>	18
2.5 <i>Illustration of an SFMCW LiDAR</i>	20
2.6 <i>Illustration of a Pulse LiDAR measurement</i>	21
2.7 <i>An example of of single-stop LiDAR method</i>	21
2.8 <i>Illustration of a single-stop LiDAR architecture</i>	22
2.9 <i>An example Multi-STOP LiDAR architecture</i>	23
2.10 <i>Illustration of Double Slit Experiment</i>	27
2.11 <i>Illustration of a HOM Dip, where 4 different outputs of 50:50 beam splitter was explained</i>	28
2.12 <i>Illustration of the beam splitter modes</i>	29
2.13 <i>Demonstration of BB84 QKD scheme [5]</i>	31
2.14 <i>Demonstration of continuous variable quantum crpytography [6]</i>	33
2.15 <i>Illustration of a FPGA Fabric</i>	34
2.16 <i>Illustration of ASIC types</i>	36
2.17 <i>Illustration of a coarse counter TDC and its waveforms</i>	39
2.18 <i>Illustration of Multi-phase counter</i>	40
2.19 <i>Multi-phase counter waveforms</i>	41

LIST OF FIGURES

2.20	<i>Waveforms of a Nutt Interpolation [7]</i>	42
2.21	<i>Illustration of Nutt interpolation architecture</i>	42
2.22	<i>Illustration of TAC based TDC</i>	43
2.23	<i>Waveforms of TAC based TDC</i>	43
2.24	<i>Time Expansion TDC</i>	45
2.25	<i>Waveforms of the Time Expansion TDC</i>	45
2.26	<i>A Tapped Delay Line TDC</i>	46
2.27	<i>Waveforms of a Tapped Delay line TDC</i>	47
2.28	<i>Demonstration of a carry chain within an FPGA SLICE [8]</i>	48
2.29	<i>A Vernier Delay Line</i>	50
2.30	<i>A Vernier Ring [1]</i>	51
2.31	<i>Multi-ring oscillator TDC [9]</i>	52
2.32	<i>Multi-ring oscillator TDC waveforms [9]</i>	52
2.33	<i>The pulse shrinking TDC [10]</i>	53
2.34	<i>The pulse shrinking TDC waveforms [10]</i>	54
2.35	<i>The Local Passive Interpolator [1]</i>	55
2.36	<i>A stochastic TDC [11]</i>	56
2.37	<i>A Successive Approximation TDC</i>	58
2.38	<i>Gated Ring Oscillators [11] [9]</i>	59
2.39	<i>Waveforms for the Algorithmic TDC</i>	60
2.40	<i>A SERDES based TDC</i>	61
2.41	<i>Illustration of type a wave-union launchers</i>	62
2.42	<i>Illustration of jump patterns</i>	63
2.43	<i>Illustration offset and gain errors of a TDC</i>	67
2.44	<i>Illustration of ideal delay line bins [12]</i>	68
2.45	<i>Illustration of non-ideal delay line bin [12]</i>	68
2.46	<i>Illustration of an ideal TDC transfer function [12]</i>	68
2.47	<i>The Non-linear vs Linear transfer function of a TDC</i>	69
2.48	<i>Illustration of missing bins in a histogram</i>	70
2.49	<i>The TDC Quantisation Error vs PDF</i>	71
2.50	<i>Illustration of flip-flop metastability</i>	73
2.51	<i>Calibrated versus non-calibrated TDC transfer functions</i>	75
2.52	<i>Illustration of the Equivalent Coding Lines</i>	76
2.53	<i>Illustration of the Double Registration TDC</i>	78
2.54	<i>Illustration of the Multi-Chain Registration TDC</i>	78
2.55	<i>Illustration of a Sliding Scale TDC [1]</i>	79
2.56	<i>Illustration of a coincidence address</i>	82
2.57	<i>The original drawings of Rossi's coincidence circuit [13]</i>	83
2.58	<i>Illustration of an analogue coincidence circuit</i>	84
2.59	<i>Illustration of height analyser coincidence detection block</i>	85

2.60	<i>Illustration of an energy levels caused by pile-up</i>	85
2.61	<i>Generation of the trigger based on the input decay</i>	86
2.62	<i>The semi-period measurement [11]</i>	86
2.63	<i>Waveforms of and gate coincidence counter</i>	87
2.64	<i>Illustration of a pulse-shaping coincidence counter</i>	88
2.65	<i>Waveforms of a pulse-shaping coincidence counter</i>	88
2.66	<i>Illustration of an edge detection coincidence counter</i>	89
2.67	<i>Waveforms of a edge detection coincidence counter</i>	89
2.68	<i>Asynchronous latch coincidence counter</i>	90
2.69	<i>Waveforms of an asynchronous latch coincidence counter</i>	91
2.70	<i>A block diagram of a TDC based coincidence counter</i>	91
2.71	<i>Illustration of a rolling window method</i>	93
2.72	<i>Illustration of a forward looking tag difference method</i>	94
2.73	<i>The coincidences detected from different channel perspectives</i>	95
2.74	<i>Operational block diagram of multi-channel labeling coincidence counter and the tag exchange between channels</i>	96
2.75	<i>The system diagram of SFQ coincidence circuit</i>	97
3.1	<i>The TDC based coincidence counter software [12]</i>	102
3.2	<i>The block diagram of backward looking tag difference coincidence counter</i>	106
3.3	<i>The system overview of tag serialising coincidence counting system</i>	108
3.4	<i>The tag generation for tag serialising TDC [7, 9]</i>	109
3.5	<i>Illustration of coincidence counting scheme [7]</i>	110
4.1	<i>TDC tag generation process</i>	114
4.2	<i>Multi-chain delay lines TDC</i>	116
4.3	<i>Multiple Times Registration TDC</i>	116
4.4	<i>Example of dual clock edge quantisation for inputs</i>	118
4.5	<i>High-level overview of logical design layout within FPGA fabric [14]</i>	119
4.6	<i>Non-linearity within delay-line structure [14]</i>	119
4.7	<i>Bin widths across delay-line</i>	121
4.8	<i>Cumulative distribution of counts across delay-line structure</i>	122
4.9	<i>Waveforms of the trigger problem when the trigger is synchronised with clock edge</i>	123
4.10	<i>Waveforms when the trigger is generated by the counter</i>	124
5.1	<i>Components of the coincidence counting system</i>	128
5.2	<i>The overview of coincidence counting system</i>	129
5.3	<i>Illustration of delta time measurement for multiple stops by the tag correla- tor [14]</i>	131
5.4	<i>Illustration of coincidence counting block</i>	133

LIST OF FIGURES

5.5	<i>The demonstration of coincidence fold detection from multiple channels . . .</i>	136
5.6	<i>The Opal Kelly XEM6310 System Components [2]</i>	137
5.7	<i>The Opal Kelly XEM6310 Board with the breakout board</i>	139
5.8	<i>Terminal output for the coincidence counter</i>	141
6.1	<i>The timing jitter test experimental setup</i>	144
6.2	<i>Timing jitter measurement with non-calibrated TDC</i>	145
6.3	<i>Timing jitter measurement with the calibrated of the TDC</i>	146
6.4	<i>Timing jitter measurement with the linearised and calibrated TDC</i>	146
6.5	<i>non-calibrated bin counts of the delay line</i>	148
6.6	<i>Calibrated bin counts of the delay line</i>	148
6.7	<i>Calibrated and linearised bin counts of the delay line</i>	149
6.8	<i>Calibrated, linearised and scaled up bin counts of the delay line with 10-bit code</i>	149
6.9	<i>The differential non-linearity before and after linearisation</i>	150
6.10	<i>The differential non-linearity before and after scaling</i>	150
6.11	<i>The integral non-linearity before and after linearisation</i>	151
6.12	<i>The integral non-linearity before and after scaling</i>	151
6.13	<i>The transfer functions for linearised, calibrated and non-calibrated codes . .</i>	153
6.14	<i>The transfer function comparison for linearised, calibrated and non-calibrated codes for early bins</i>	153
6.15	<i>The transfer function comparison for linearised, calibrated and non-calibrated codes for late bins</i>	154
6.16	<i>The transfer function comparison between linearised and only calibrated codes</i>	154
6.17	<i>Histogram of 5 STOPS computed by the correlator [14]</i>	155
6.18	<i>Histogram of 43 STOPS computed by the correlator [14]</i>	156
6.19	<i>Histogram of 120 STOPS computed by the correlator [14]</i>	156
7.1	<i>The coincidence rate as a function of the delay with window size 380 ps - 7.7 ps</i>	160
7.2	<i>The coincidence rate as a function of the delay with window size 770 ps - 9.9ns [3]</i>	161
7.3	<i>Coincidence rates (singles and 2-folds) vs input frequency [3]</i>	162
7.4	<i>The coincidence rate as a function of the delay measured with Picoharp300 .</i>	164
7.5	<i>The coincidence rate as a function of the delay measured with the coincidence system</i>	164
7.6	<i>Quantum photonics experimental benchmarks of the counting system in quantum interference experiment [3]</i>	165
7.7	<i>The detectors' delay calibration with respect to the reference channel (channel 1) [3]</i>	166

LIST OF FIGURES

7.8	<i>Quantum photonics experimental benchmarks of the coincidence counting system in a pseudo-photon-number resolving detection of a coherent state of light [3]</i>	167
8.1	<i>Multi-phase registration TDC Waveforms</i>	176
8.2	<i>Multi-phase registration TDC architecture</i>	177
8.3	<i>Dual Data Rate TDC architecture</i>	178
8.4	<i>Dual Data Rate TDC Waveform</i>	178
1	<i>The coincidence rate as a function of delay with Rolling Window [12]</i>	204
2	<i>The coincidence rate as function of delay with Forward Looking Tag Difference method [12]</i>	204
3	<i>The coincidence rate backward looking tag difference method [11]</i>	205
4	<i>The coincidence rate backward looking tag difference method [11]</i>	205
5	<i>The coincidence rate as a function of the delay [7]</i>	206
6	<i>The coincidence rate as a function of the delay [7]</i>	206

List of Acroynms

ADC	Analogue-to-Digital Converter
AM	Amplitude Modulation
AMCW	Amplitude Modulated Continuous Wave
APD	Avalanche Photo-Diode
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
BPSK	Binary Phase Shift Keying
BRAM	Block Random Access Memory
CDF	Cumulative Density Function
CIN	Carry-In
CMOS	Complementary Metal-oxide-semiconductor
COUT	Carry-Out
CPLD	Complex Programmable Logic Device
CPS	Cyclic Pulse Shrinking
CW	Continuous Wave
DAC	Digital-to-Analogue Converter
DCM	Digital Clock Manager
DDR	Dual Data Rate
DNL	Differential Non-Linearity
DSP	Digital Signal Processor
ECL	Equivalent Coding Lines

LIST OF FIGURES

FIFO First-in First-out

FMCW Frequency-Modulated-Continuous Wave

FPGA Field Programmable Gate Array

FSR Finite Step Response

FWHM Full width at half maximum

GM Geiger-Muller

GRO Gated Ring Oscillators

HOM Hong-ou-Mandel

I/O Input/Output

IC Integrated Circuit

IIR Infinite impulse response

InGaAs Indium Gallium Arsenide

INL Integral Non-Linearity

ISR Infinite Step Response

JTL Josephson Transmission Lines

LiDAR Light Detection and Ranging

LPI Local Passive Interpolator

LSB Least Significant Bit

LUT Look-up Tables

LVDS Low-voltage Differential Signaling

MB Mega Byte

MC-DC/SFQ Magnetic Coupled DC/SFQ

MCPS Million Counts Per Seconds

MiB Million Bits

MPC Multi-Phase Clock

MTBF Mean Time Between Failures

ODDR Output Dual Data Rate

PC Personal Computer

PCB Printed Circuit Board

PCI Peripheral Component Interconnect

PCXD Photon Counting X-Ray Detectors

PDF Probability Density Function

PECL Positive Emitter-Coupled Logic

PET Positron Emission Tomography

PLL Phased-Locked Loop

PNRD Photon Number Resolving Detector

PS Pulse Shrinking

PSoC Programmable System-on-Chip

QKD Quantum Key Distribution

RADAR Radio Detection and Ranging

RAM Random Access Memory

SA Successive Approximation

SBS Scattershot Boson Sampling

SDRAM Synchronous Dynamic Random Access Memory

SERDES Serialiser/Deserialiser

SFMCW Stepped Frequency Modulated Continuous Wave

SFQ Single Flux Quantum

SMA Sub-Miniature A

SNDR Signal-to-Noise and Distortion Ratio

SNR Signal-to-Noise Ratio

SNSPD Superconducting Nanowire Single-photon Detectors

SoC System on a Chip

LIST OF FIGURES

SONAR Sound Navigation Ranging

SPAD Single Photon Avalanche Detector

SPD Single-Photon Detector

SRAM Static Random Access Memory

SSP Single-Shot Precision

STD Standard Deviation

TAC Time-to-Analogue Converter

TCL Timing Coding Lines

TDC Time-to-Digital Converter

TE Time Expansion

ToF Time-of-Flight

USB Universal Serial Bus

VCRO Voltage-Controlled Ring Oscillators

VDL Vernier Delay Line

VHDL VHSIC Hardware Description Language

VHSIC Very High-Speed Integrated Circuit

VOA Variable Optical Attenuator

Nomenclature

ΔE	The change in energy
ΔF	The frequency difference between events
ΔT	The time difference between events
ϕ	The phase between the transmitted and reflected signals
π	3.14159
ψ	The bosonic wave function
σ	The standard deviation
θ	The extra phase shift added due to the offset phase shift
C	Speed of light in a vacuum, 299,792,458m/s
erf	The Gaussian error function
f	The frequency
f_d	The doppler frequency
H	The transfer function of a converter
h	Planck's constant, 6.626×10^{-34}
t	Time
T_{clk}	The clock period
T_{diff}	The time difference between the multi-phase clocks in multi-phase counter TDC
T_d	The delay on a single delay element
T_{event}	The time of event
T_{gh}	The upper bound of the coincidence window

LIST OF FIGURES

T_{gl} The lower bound of the coincidence window

T_{res} The resolution of the TDC

T_{START} The time of the START event

T_{STOP} The time of the STOP event

T_w The coincidence window size

w Angular frequency

$\hat{a}_n^\dagger, \hat{b}_m^\dagger$ The bosonic creation operators for beam splitters modes a and b

Publications

Conference Papers :

- E. Arabul, A. Girach, J. Rarity, and N. Dahnoun, “Precise multi-channel timing analysis system for multi-stop lidar correlation,” *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2017
- E. Arabul, J. Rarity, and N. Dahnoun, “Fpga based fast integrated real-time multi coincidence counter using a time-to-digital converter,” *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, 2018
- S. Tancock, E. Arabul, N. Dahnoun, and S. Mehmood, “Can dsp48a1 adders be used for high-resolution delay generation?,” in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–6, June 2018

Journal Papers :

- S. Tancock, E. Arabul, and N. Dahnoun, “A review of new time-to-digital conversion techniques,” *IEEE Transactions on Instrumentation and Measurement*, vol. PP, pp. 1–1, 08 2019
- E. Arabul, S. Paesani, S. Tancock, J. Rarity, and N. Dahnoun, “A precise high count-rate fpga based multi-channel coincidence counting system for quantum photonics applications,” *IEEE Photonics Journal*, vol. 12, no. 2, pp. 1–14, 2020

Introduction

Precise time measurement and correlation are essential concepts in many engineering and science applications. Coincidence counters can be an example of one of the correlation tools, which are used in almost every quantum photonics applications. Coincidence counter applications include photonic quantum simulators [15], quantum communication experiments [16] and boson sampling machines [17]. Their purpose in a photonic setup is measuring the correlation between simultaneous trigger events generated by Single-Photon Detectors (SPDs). The real-time processing of such events is of increasing importance as quantum photonics experiments increase in complexity, where the total number of events becomes too large to be stored and post-processed [18, 3].

For instance, in quantum computer problems such as Scattershot Boson Sampling (SBS), the theoretical efficiency of spontaneous generation of n number of photon pairs exponentially decreases as the n number increases. [19, 18]. As it was demonstrated in [18], generated 100 million single photon events per second only yields to 4 8-photon events per hour. Therefore, the processing a vast number of time events is necessary for expanding the capabilities of the quantum computer. Since the sequential calculation of the correlation between millions of time events with each other is exhaustive for a modern CPU, sequential post-processing based coincidence counting never achieves high count rates in a quantum application. A typical performance of a sequential post-processing can be seen in [12], where 2 million events could be processed in a second. With a combination of multi-threading and data compression techniques, software based post-processing achieves a measurement of 40 million events per second in PicoQuant QuCoa [20] and 65 million events per second achieved with Swabian Instruments Time Tagger Ultra [21]. Although, these instruments are the

best post-processing based coincidence software available, they would still bottleneck the amount quantum information can be processed since a typical SPDs system can provide count rates up to 200 Million Counts Per Seconds (MCPS) [22]. Hence, it is essential for the coincidence instrument to work beyond the count rates of detectors for not being the limiting factor in quantum experiments.

Moreover, coincidence counting allows experiments to filter out the background noise (uncorrelated photons and dark counts) from the entangled photons generated and processed in quantum experiments. An illustration of coincidence counting in a quantum setup can be seen in Figure 1.1. Other than quantum photonics, coincidence counters are found in applications including image reconstruction through the gamma-ray correlation in Positron Emission Tomography (PET) [23] and Neutron Detectors in Nuclear Chemistry [24]. However, coincidence counters will be only discussed as an instrument used in quantum photonics applications in this thesis.

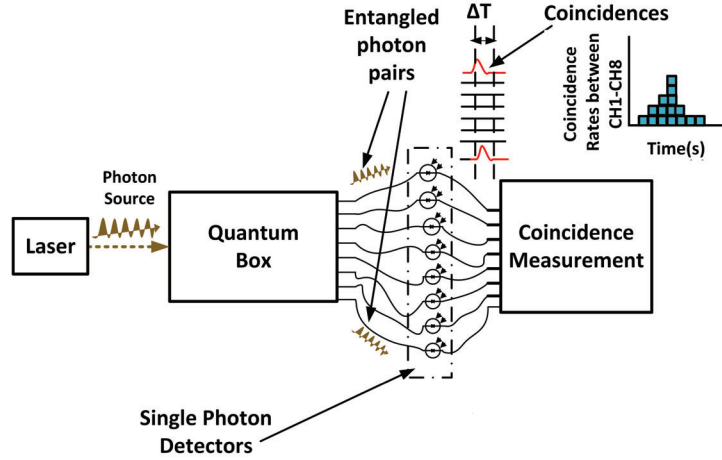


Figure 1.1: *The example usage of a coincidence measurement in a quantum application [3]*

To implement a precise and flexible coincidence counter, Time-to-Digital Converters (TDCs) can be used. TDCs are used for generating timestamps for the events occurring in the time domain. Traditionally, TDCs are used for measuring the time difference between the START and STOP events. However, generally in digital implementations such as the ones using Field Programmable Gate Array (FPGA), the STOP signal is replaced by the system's clock edge, and inputs' positions are measured relative to the clock edge [1]. For the fine time measurement, asynchronous logic circuits such as carry chains are used to subdivide the clock period into the smaller bins, and in quantum photonics applications, TDCs are employed for digitising the time of arrival of photons [25, 3].

The phenomenon of coincidence is defined as two or more events occurring within

a time interval simultaneously [26]. This time interval is known as the coincidence window. In quantum photonics tasks, these coincidences can occur across multiple channels and with different combinations [27, 28]. Coincidence counters are employed to measure these patterns and count them. Coincidence counter uses these patterns internally as addresses, with associated memory locations incremented when patterns are detected [3].

The introduction chapter will provide a brief background about similar implementations present, aims and constraints of this research, research contributions and the thesis outline. The chapter will continue with a background in the next section.

1.1 Background

Developments in single-photon detectors promise to achieve significantly higher count-rates [29] in quantum photonics applications. However, timing measurement tools are struggling to keep up with data-rates of the detectors, especially in high precision and multi-channel instruments. Popular commercial coincidence counter tools available include ID Quantique’s ID900, which can count with 4 channels at 25 MCPS per channel (with a total of 100 MCPS) while the Single-Shot Precision (SSP) was at 8 ps [30], and PicoQuant’s HydraHarp400 [20] with a software-based coincidence counter. The maximum count-rate achieved by the HydraHarp was 12.5 MCPS per channel and 40 sustained MCPS across 8 channels [3]. Also, Swabian Instruments Time Tagger Ultra achieves 17 channels with 65 MCPS where the precision is around 10 ps[21].

In the literature, some attempts were previously made at implementing multi-channel coincidence counters. This includes a coincidence counter using TDCs implemented to use for Time-of-Flight (ToF) PET cameras where they achieved a total count-rate of 72 MCPS, and the resolution was at 69.8 ps [31]. This implementation used a separate 12 channel TDC board to achieve the coincidence counting, which hindered the count-rate of the operation. Another plausible coincidence instrument in the literature is a vast scale 32-channel coincidence counter implementation for the Boson Sampling [32]. Their method achieved 390 ps resolution by using an octa-phase TDC. Although the scale of the implementation was impressive, due to the usage of a large FIFO, the count-rate of the operation was limited to 0.5 MCPS. In addition, [33] demonstrates another very large scale 64 channel TDC based coincidence counter for animal PET scanners, which was implemented in a Xilinx Virtex 2 FPGA. This implementation used an external Analogue-to-Digital Converter (ADC) board for the time tag generation and achieved 0.7 ns resolution with a 32 MCPS data-rate [3].

Although TDCs have been used for coincidence counting for a while, the main problem that hinders the data-rate of the coincidence counting is the data transfer between the TDC and coincidence counter since data cannot be processed as it is generated. The work presented in this thesis proposes a coincidence counting system that integrates

the multi-channel coincidence counting and high-precision time tag generation into the same FPGA fabric, which mitigates the use of any data transfer protocol between the TDC and coincidence counter. Implementation techniques, concepts and the further background will be discussed in Chapter 2. Aims and constraints of this research will be discussed in the following section.

1.2 Aims & Constraints

The proposed real-time coincidence counting system should provide certain functions to be able to be used in quantum photonics applications. Aimed operations for the proposed system could be listed as below :

- Precisely measuring the timing of events (sub 10 ps precision).
- Flexibly changeable coincidence window sizes and digital delays on each channel.
- Detecting coincidences across all channels and forming coincidence patterns.
- Counting occurrence of coincidence patterns in real-time (typically in a 50-500 ms integration time).
- Displaying results of the coincidence counting after each integration time.
- Providing an optimal count-rate while providing a real-time operation (30 > MCPS for a single channel, i.e 240 MCPS for 8 channels).

These aims were constructed for being compatible with the needs of quantum experiments. Some of these aims were chosen based on detector technology, such as precision and count-rate. Other aims were decided for the experimental needs during the coincidence counting operation. The significance of the aimed precision and count-rate is the instrument's compatibility with the most advanced Superconducting Nanowire Single-photon Detectors (SNSPD) systems. As the topical review (2019) conducted on state-of-art SPDs used in the quantum applications, [34] stated that the current superconducting nanowire technology achieves 32.3 ps timing jitter (Full width at half maximum (FWHM)) which yields to around 13 ps SSP. Also, in another review (2020) [35], currently the superconducting nanowire technology started to show timing jitter of 15 ps which corresponds to around 7 ps SSP. Hence, aiming for sub 10 ps precision is a realistic goal for the coincidence detection system to be compatible with the advances in the detector technology. In terms of count rates, as it was stated in [35], the highest achieved count rate for a single detector is at 30 MCPS which corresponds to 240 MCPS for an 8 channel system. Thus, to surpass the detection rates, a coincidence counter should aim to provide count rates over 30 MCPS for a single channel.

The configurable window and channel delays, adjustable integration times and displaying results after each integration time are required specifications for a flexible coincidence counting operation. The flexibility is required since the instrument can be used in many unique experiments where detectors with different specifications and varied lengthen optical wires can be used. Hence, the instrument can be adjusted accordingly. An example of flexible coincidence counter can be seen in [36] where multi-channel, configurable windows and adjustable integration time are provided.

To achieve these aims, the proposed TDC architecture can be found in Chapter 4 and for the coincidence counter architecture Chapter 5 should be referred. The precision of the proposed TDC can be found in Chapter 6 and the performance of the coincidence counter can be seen in Chapter 7.

While achieving these operations, coincidence counter implementation needs to address 3 main constraints; the timing precision, maintenance of the real-time and data transfer throughput.

1.2.1 Timing Precision

Precision and accuracy are commonly two misunderstood concepts. While the accuracy represents the closeness of a measurement to the true value of the measured quantity, the precision is how repeatable measurements are and the agreement between repeated measured values [37]. In the concept of the timing measurement, the accuracy is observed as a constant offset delay and typically affected by routing and wire delays present in the setup. However, the timing precision is the main constraint in the timing measurement, and it defines the performance of the timing instrument by determining the error margin affecting each measurement. This error margin can be expressed by the standard deviation (σ), where it is shown with \pm alongside the measured value. Also, a SSP is used to express the precision of the instrument, which is $\frac{\sigma}{\sqrt{2}}$. Although there are plenty of different instruments with different precision, typically high precision devices used in quantum photonics aims to provide around 10 ps SSP. TDC architectures are generally the core of the timing instrumentation, and the research of the timing measurement focuses on improving the precision of TDC architectures. TDC schemes used for this purpose will be discussed in Chapter 2, and the proposed implementation in Chapter 4. The achieved precision can be found in Chapter 6. The difference between the timing accuracy and precision is illustrated in Figure 1.2, where accuracy represents the offset affecting mean of measurements while the precision is the distribution of results.

Another issue is needed to be addressed for a timing instrument is the linearity problem. The linearity of a TDC is known as the deviation of TDC's transfer function from an ideal. Non-linearity directly affects the precision of a TDC, and it is usually caused by power fluctuations, temperature fluctuations and internal routing of the

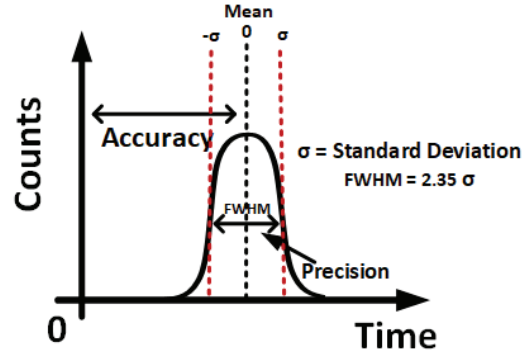


Figure 1.2: *Illustration of timing accuracy and precision*

converter. To improve the linearity, typically calibration methods and linearisation methods are utilised. These methods will be discussed in Chapter 2.

1.2.2 Maintaining Real-Time Operation

The real-time processing of time events for coincidence counting has become important due to an increase in the complexity of the quantum photonics experiment. The real-time operation from quantum photonics point of view can be described as continuous processing of time events as they are being generated during an experiment. This is typically done for an integration time, whose duration is specified for the application. During the integration time, time events are generated and processed. Also, at the end of an integration time, results are output, and the system is reset. The integration time can be typically in a range of microseconds to seconds.

There are mainly three places where the real-time operating TDC based coincidence counter can be constrained. The first place is the dead-time of the TDC, which is the required time between measurements for a system to settle for the next measurement. Secondly, interconnects between the TDC and coincidence counter. Interconnects are especially the case when the TDC and coincidence counter needs to transfer data between different chips or systems. The last place is the operation of the coincidence counting, where forming and storing coincidence patterns can introduce additional dead-times. The places where real-time operation can be interrupted can be seen in Figure 1.3.

The dead-time of a TDC is typically introduced in three different places in the system. The first source of dead-time is introduced when quantising the trigger event, which is generally unavoidable and limited by the system's clock period since it is necessary to synchronise events with the system's clock to be able to process them in the system. The second place is when the design is not pipelined, and Random Access Memory (RAM) read and write operations are handled inefficiently, and this

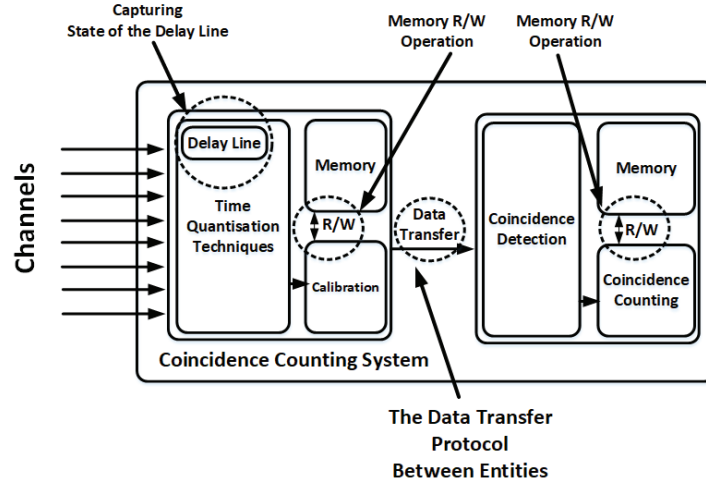


Figure 1.3: *Places where the real-time operation can be interrupted in a coincidence counting system*

can result in an additional number of clock cycles during the calibration of a TDC if calibration is required. The third place is how the system registers input triggers, and thus, consecutive trigger signals can be missed by the system if they are not handled correctly. The proposed TDC scheme for addressing these constraints can be seen in Chapter 4.

Interconnects can also be a problem since the data transfer between the TDC, and coincidence counter might be relying on a data serialisation of parallel generated data. The data serialisation is typically done by using a FIFO and for each additional number of operational channels will result in an additional number of clock cycle to the dead-time. An example of data serialisation using TDC based coincidence counters can be seen in Chapter 3 and the proposed architecture which tackles this issue can be found in Chapter 5.

The final place where real-time performance is affected is the coincidence counting operation. Using a memory block (typically a RAM) for the coincidence counting makes it inevitable to have at least two clock periods of dead-time since stored counter values in the memory is needed to be read first and written later in order to be incremented. The details of this operation can be seen in Chapter 5 and in Chapter 7 impacts of the RAM operation on a real-time operation will be discussed.

1.2.3 Data Transfer Throughput

The data transfer throughput is generally the main constraint that limits the count-rate of the TDC based coincidence counting instrument. This happens due to a need for generated time tags to be transferred to the coincidence counting unit by using a

data transfer protocol. When the TDC and coincidence counter are implemented in different platforms, these protocols become essential. For this purpose protocols such as Universal Serial Bus (USB) 2.0 , USB 3.0 or Peripheral Component Interconnect (PCI) Express etc. can be utilised. However, regardless of how fast the protocol's data-rate is, these protocols are always limiting factors for the TDC based coincidence counting since there is a different data-rate threshold that is enforced by each protocol. These protocols will effectively limit the available number of operational channels, a number of bits can be in a time tag (the precision and range) or simply the high count-rate detection. In Chapter 3, a software-based coincidence counter implementation using a TDC, which suffers from this issue, can be found.

An illustration of the data throughput problem can be seen in Figure 1.4, where Figure 1.4a shows the data serialisation process between the data generation and data processing, and Figure 1.4b shows the comparison of generated and transferred data with the data-rate saturation of the data transfer protocol.

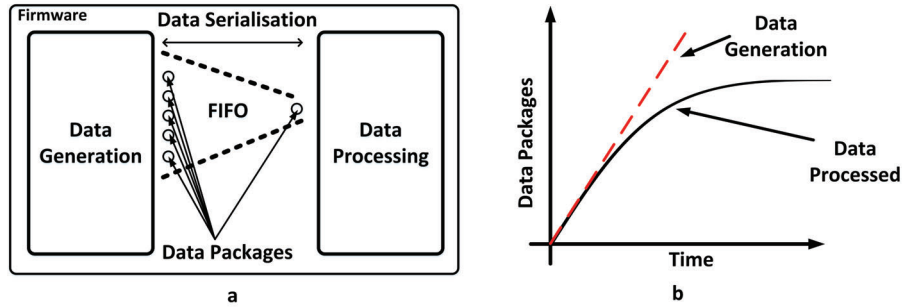


Figure 1.4: *The Illustration of the data transfer throughput, a : The Data Serialisation between data generation and processing, b: The graph of transferred and generated data*

In order to overcome the data transfer throughput constraint, the TDC and coincidence counter can be integrated into the same fabric to avoid all the data transfer protocols between them. Therefore, the optimal count-rate for the coincidence counter can be achieved. The section 2.6.5.3 in Chapter 2 provides an implementation scheme, which can be used to overcome this constraint. How the proposed system integrates a TDC with a coincidence counter can be found in Chapter 5 and the system's count-rate performance can be found in section 7.2.2 in Chapter 7.

1.3 Thesis Outline

Chapter 2 - Research Review : This chapter provides fundamental concepts to understand the work presented in the thesis. This includes essential LiDAR techniques used in ToF measurements, a brief introduction to the Quantum Information Theory

and Quantum Interference, a comprehensive TDC and Coincidence Counter research review.

Chapter 3 - Legacy Coincidence Counters : This chapter covers previously designed coincidence counter implementations by the author before the final coincidence counting system. These implementations include the software-based coincidence counter using a TDC, FPGA based Backward and Forward Looking Tag Difference coincidence counters. The results achieved from these implementations are recapped in Appendix 8.3.3.

Chapter 4 - Time-to-Digital Converter Implementation : This chapter provides the implementation details behind the TDC design, which includes how self-calibration modules and Dual Data-Rate Registration method were implemented within the FPGA. Also, the theory behind averaging methods and the code density testing are being expanded in this chapter.

Chapter 5 - A Precise High Count-Rate Multi-Channel Coincidence Counting System : This chapter discusses the implementation details of the coincidence counting system, the hardware and software interfaces used in the implementation. The modules such multi-tag correlator, coincidence detector and coincidence counter are discussed in this section. Also, the software part where the USB 3.0 interface and post-processing were discussed.

Chapter 6 - Functionality, Linearity and Precision of the Time-to-Digital Converter : This chapter aims to discuss the performance of the TDC in terms of functionality, linearity and precision. The functionality was discussed in terms of the capability of detecting an arbitrary number of STOPS between two STARTs. The precision was discussed with parameters of SSP, Standard Deviation (STD) and FWHM. The linearity was discussed in terms of bin widths, the Integral Non-Linearity (INL) and Differential Non-Linearity (DNL).

Chapter 7 - Coincidence Counting Benchmarks and Applications : This chapter presents the results achieved by the proposed coincidence counting system in certain test scenarios. These tests showed the minimum precise window size to the highest count-rate achieved by the instrument. Also, the results achieved from an actual optical setup, which include its performance comparison with PicoHarp300, the rev-HOM dip measurement in two single-photon interference and the pseudo-photon-number resolving detection of a coherent state of light.

Chapter 8 - Conclusion : This chapter summarises the work presented in the thesis and provides future work for the research. These future works include how to improve the precision, count-rate and channel numbers.

1.4 Research Contributions

This thesis presents the following contributions to the research in the area of the timing measurement instrumentation and the coincidence counting in quantum photonics. After research contributions are presented, the thesis will continue with Chapter 2: Research Review.

Precise Timing Instrument :

- A precise TDC was implemented in a Spartan 6 LX150 FPGA chip and a delay line based TDC achieved 8.9 ps SSP where the Least Significant Bit (LSB) resolution was 7.7 ps with max DNL error of 2.9 LSB and max INL error of 8.8 LSB. This TDC was used for implementing the work presented in [3]. This can be seen in Chapters 4, 6.
- A code density calibration based on-the-fly TDC calibration was integrated into the FPGA fabric for a Multi-STOP LiDAR correlation system, which was published in [14] and covered in Chapter 4 and Chapter 6.
- The Dual Data Rate Registration TDC scheme has been developed to improve the linearity and precision of the TDC without using additional delay lines and adding dead-time [3]. The scheme is covered in Chapter 4 and Chapter 7. This method implemented an averaging TDC method using both clock edges in combination with the code density calibration method. With this method 1.2 LSB in max DNL error, 1.8 LSB in max INL error, 10 ps in FWHM and 3.2 ps in SSP improvements were achieved.
- A Multi-Stop LiDAR instrument to achieve a multi-stop time differencing method for a time correlation operation, which is not affected by uncorrelated detection of STOP signals, has been developed. For this method, Chapter 2 4 and 6 can be referred. Also, it was published in [14].
- Contributions to the surveillance and discussion of new TDC techniques has been done, which was published in [1] also covered in Chapter 2.

Multi-Channel High Count-Rate Coincidence Counting System :

- The development of a multi-channel labelling coincidence counting system which utilises the multi-stop LiDAR method and delay line TDC. The coincidence counting system is unique in a way that it combines a TDC with a coincidence counter into the same FPGA fabric and providing coincidence counting concurrently for each channel. This method achieved the highest count-rate of 320 MCPS (40 MCPS per channel) for a precise (around 10 ps SSP) multi-channel (> 8 channels) timing analyser instrument. This work is presented in [3] and in Chapters 5, 7.
- The system's suitability for Quantum Photonics experiments was demonstrated in applications such as performance comparison with PicoHarp300, reverse Hong-ou-Mandel (HOM) dip measurement in two single-photon interference and pseudo-photon-number resolving detection of a coherent state of light. For the results obtained, Chapter 7 and [3] can be referred.
- The coincidence counter showed the capability of detecting all the generated coincidences with 107 ps window size and the smallest usable coincidence counter was 7.7 ps. This can be seen in Chapter 7 and in [3]. Also, in the first iteration of this implementation, 150 ps window size was managed to capture all the generated coincidences as it was published in [7]. The legacy of the proposed system can be found in Chapter 3.

CHAPTER 2

Research Review

2.1 Introduction

In this chapter of the thesis, the fundamental knowledge required to understand the work presented in this thesis will be presented. Background about the concept of time-of-flight (ToF) measurements and techniques involved in implementing time measurement and coincidence counting instruments will be discussed. Since the thesis is focusing on the time measurement in quantum photonics applications, these topics will be discussed from the perspective of measurement of photons' time-of-arrivals and their coincidence correlations. This chapter will start with an introduction to time-of-flight measurements and followed by the LiDAR overview and Quantum Information. After these concepts are covered, the rest of the section will focus on TDC and coincidence counting techniques in the literature.

2.2 Time-of-Flight Measurements

With developments in electronics, measuring the distance using wave propagations has become popular in science and engineering applications. Time-of-Flight can be defined as imaging and ranging techniques use the time that it takes waves to travel from the source to the target and back to the receiver after being reflected from the target. Depending on the needs of the application, different waves types can be chosen, waves' propagation mediums affect the propagation speed of the waves during an operation. Typically, when radio signals are used it is named as Radio Detection and Ranging (RADAR), when ultrasonic waves are used as Sound Navigation Ranging

CHAPTER 2. RESEARCH REVIEW

(SONAR) and when light waves are used for the ranging it is named as Light Detection and Ranging (LiDAR).

The earliest form of Time-of-flight measurement instruments were probably ultrasonic pulse measurement devices, where the sound waves were used for measuring the distance between the object and source. Using a pulse generator is probably the simplest ranging technique, where reflected pulses from a target are used for the measurement. However, the wave modulation is also commonly used for ToF measurements, where the phase difference between the emitted event and reflected event is compared for the distance measurement.

ToF systems can be found in many different applications includes automotive industry [38, 39], agriculture [40], civil engineering [41], mining [42] and quantum information [43] etc. In this thesis, since it was written from the quantum photonics' perspective, the main focus will be the LiDAR measurement. In the following section, the LiDAR overview and common LiDAR techniques used today will be discussed. LiDAR Techniques can be also used to investigate the time correlation between photons. Therefore, LiDAR principles are needed to be covered to understand how ToF of photons can be measured in quantum photonics applications.

2.2.1 LiDAR Overview

LiDAR is a distance measurement technique, which uses the time-of-flight of photons. This is done by emitting light from a source to illuminate the target, and once, the light is reflected from the target, the reflected photons are detected by a detector [44]. The time between the emission of light and detection of reflected photons is used for measuring the distance. The moment the light source starts emitting photons is denoted as the START event and the time the reflections are detected by the detector is defined as the STOP. Since the speed of light (C) is a known value, from the time of flight the distance between the transmitter and the target can be measured [4].

There are typically two types of LiDAR approaches; Pulse LiDARs and Modulation LiDARs. Modulation LiDARs typically use Continuous Wave (CW) lasers to send waves to a target and measures the phase difference between the transmission and detection of the reflection. Pulse LiDARs use pulse lasers for rapid transmission of pulses to a target and collecting returns for measuring the time between the transmission and the detection. The principles of implementing a LiDAR is also very similar to the other imaging devices such as RADAR and SONAR. The main difference between these methods are the medium and wave types are used for the range finding [4].

2.2.1.1 Modulation LiDAR

Modulation based ToF cameras conventionally utilise the wave modulated light to illuminate the scenery. The distance that is travelled by light is $2x$, where x is the

2.2. TIME-OF-FLIGHT MEASUREMENTS

distance from the camera to the illumination target. The time of flight introduces a phase shift upon detection by cameras to amplitude modulation. This phase shift of ϕ can be defined as below, where w is the angular velocity, x is the distance, C is the speed of light [4].

$$\phi(w, x) = \frac{2wx}{C} \quad (2.1)$$

The ideal modulation of the signal detected by the camera for a pixel can be formulated below, where $\alpha(t)$ is the amplitude of the received light as a function of the time t and β is the background light [45].

$$f(t) = \alpha(t)\sin(wt + \phi(w, x)) + \beta \quad (2.2)$$

The modulation of this reference for i th pixel r_i can be expressed as below, where θ_i is the extra phase shift added due to the affecting offset phase shift and left unchanged during the calculation [45].

$$r_i(t) = \frac{1}{2}\sin(wt + \theta_i) + \frac{1}{2} \quad (2.3)$$

Amplitude at a single i th pixel K captured during the integration time of L can be formulated as below, where L is greater than the period of the modulation frequency, m_i is the amplitude and the n is the background light [45].

$$K_i(t) = \int_0^L f(t)r_i(t)dt \quad (2.4)$$

$$K_i(t) = \frac{1}{2} \int_0^L \alpha(t)\sin(\phi(w, x) - \theta_i)dt + c \quad (2.5)$$

$$K_i(t) = m_i \cos(\phi_i - \theta_i) + n \quad (2.6)$$

The phase difference (ϕ) between transmitted and reflected waves can be measured by using capacitors and an ADC. This measurement can be computed by enabling the capacitor for the detection when the signal is high and the value of the capacitor is digitised by the ADC. By subtracting the digital values for the transmitted and reflected signals, the phase difference between the transmitted and reflected waves can be found. The relationship between the phase and distance can be seen in Figure 2.1.

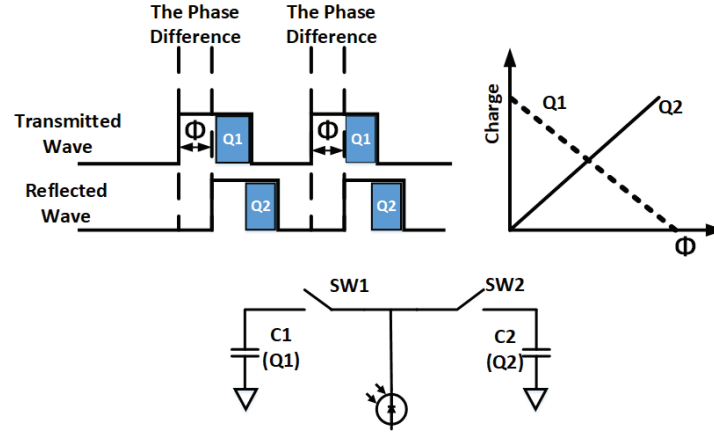


Figure 2.1: *Demonstration of the phase difference between waves and the relationship between the phase and charge stored in capacitors [4]*

Typically, there are three approaches of wave modulations that could be applied to the ToF cameras. These modulation schemes can be listed as the Frequency-Modulated-Continuous Wave (FMCW), Amplitude Modulated Continuous Wave (AMCW) and Stepped Frequency Modulated Continuous Wave (SFM CW).

Frequency Modulated Continuous Wave (FMCW) : Frequency modulation can be also used for LiDAR implementation. FMCW is based on continuously sending a signal to a target while varying the frequency, where the change in the frequency is precisely known. Thus, based on the received frequency, the time of transmission for certain frequencies can be predicted and the time of flight can be measured. The measured range of the FMCW LiDAR can be defined by the difference between the transmitted and received signals' frequency and formulated as below [45].

$$D = \frac{C|\Delta T|}{2} = \frac{C|\Delta f|}{2df/dt} \quad (2.7)$$

In Equation 2.7 C stands for the speed of light, ΔT is the period difference between two signals and Δf is the difference between the frequencies, df/dt is the frequency shift in time and D is the distance between the detector and target [46].

In this approach, the light's frequency is being increased and decreased periodically, and therefore, the distance has a relationship with the frequency range. The bandwidth of the frequency modulation determines the resolution and maximum measurable range of a LiDAR. Thus, the resolution of a LiDAR is equal to the change in the frequency within the time T . Hence, the resolution can be formulated as below.

$$T_{res} = \frac{1}{T} = \frac{df}{dt(F_{up} - F_{down})} \quad (2.8)$$

2.2. TIME-OF-FLIGHT MEASUREMENTS

In Equation 2.8, df/dt is the steepness of the frequency deviation, F_{up} is the upper frequency in the bandwidth and F_{down} is the lower frequency in the bandwidth [46].

It should be noted that if the target is moving, ΔT is also affected by the Doppler effect. Thus, ΔT includes the Doppler frequency f_d . Depending on the direction of the movement and type of the modulation signal, the Doppler frequency is dealt with accordingly. For instance, when the sawtooth modulation is used, the Doppler frequency can be ignored due to a very long measurement range, which makes the Doppler effect negligible in terms of measurement errors [46]. Illustration of FMCW can be seen in Figure 2.2.

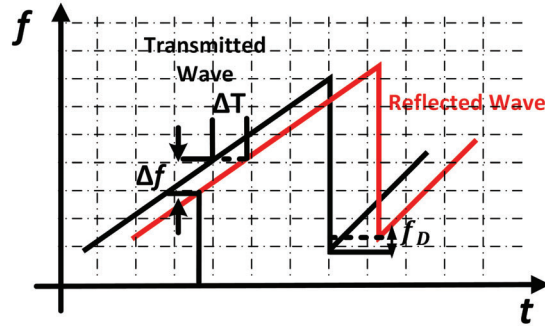


Figure 2.2: Illustration of a Sawtooth Frequency Modulated Carrier Wave LiDAR

In a triangular signal modulation, the Doppler signal can be observed as an additional height on the received signal's frequency. When there is no Doppler frequency affecting the system, the measured height of the reflection will be the same both at the rising and the falling edge. However, the Doppler frequency will cause a shift in the height of the frequency on the reflection, and hence, there will be a height difference between the measured frequency's height at both edges. The Doppler frequency can be calculated by calculating the absolute frequency difference at the rising and the falling edge and subtracting them from each other. The formula for calculating the Doppler frequency can be given as below, where f_1 is the frequency at the rising edge and f_2 is the frequency at the falling edge [46]. A diagram for triangular wave LiDAR can be seen in Figure 2.3.

$$f_d = \frac{[\Delta f_1 - \Delta f_2]}{2} \quad (2.9)$$

If the Doppler frequency is detected at the rising edge then it is added to ΔT , and for the falling edge it is subtracted from ΔT . The square wave modulation can be also used for the FMCW Lidar. However, it can only be used for very short ranges since multiple echo signals cannot be distinguished easily with this type of waves [46].

FMCW LiDAR is probably the most popular modulation LiDAR implementation today. Therefore, there are plenty of examples in the literature about this scheme.

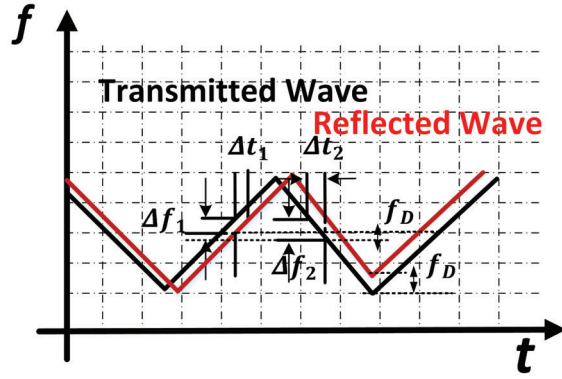


Figure 2.3: *Illustration of a Triangle Wave Frequency Modulated Carrier Wave LiDAR*

For example, as it was described in [47], with using 1 THz at approximately 1 kHz FMCW modulation, a high-precision LiDAR was implemented, which achieved $150 \mu\text{m}$ resolution with a $1 \mu\text{m}$ precision. Another example can be [48], where a 3-D scanner FMCW LiDAR was implemented using an electro-optical chip, which achieved $8 \mu\text{m}$.

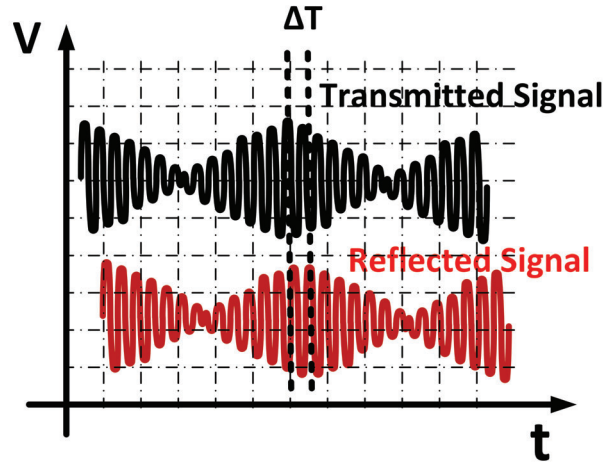


Figure 2.4: *Illustration of an AMCW LiDAR*

Amplitude Modulated Continuous Wave (AMCW) : Another variation of the modulation based LiDAR is the AMCW LiDAR. As it can be predicted from its self-explanatory name, instead of frequency modulation, amplitude modulation can be used to implement a ToF camera. This is particularly advantageous when the bandwidth of the sensors is limited, and so, the frequency is kept constant during operation. AMCW LiDAR keeps the frequency of modulation constant during the transmission and periodically change the amplitude of the frequency. Based on the amplitude of

2.2. TIME-OF-FLIGHT MEASUREMENTS

the reflected signal the time of flight can be measured. A waveform example of an AM Modulated LiDAR can be seen in Figure 2.4 [45].

When the increase in the amplitude of a reference signal is constant and equals to the maximum of 2π , reference signals for the i th pixel can be formulated as below.

$$r_i = \frac{2\pi i}{M}, \text{ where } i = 0, 1, \dots, M-1. \quad (2.10)$$

Hence, the amplitude modulation of a signal at pixel i can be expressed as below, where ϕ is the phase and n is the offset.

$$f_i = m \cos(\phi - \frac{2\pi i}{M}) + n \quad (2.11)$$

One of the examples of the AMCW in the literature can be seen in [49], where it was used for implementing a 3-D scanner. In this implementation, a polarisation insensitive LiDAR was researched and the longitude resolution of $50 \mu\text{m}$ was achieved. Also as it was described in [50], Amplitude Modulation (AM) LiDAR was used for photon counting by using 1.5 GHz sine wave Indium Gallium Arsenide (InGaAs) Avalanche Photo-Diodes (APDs). This method aimed to provide precise results in daylight and achieved 0.12 m precision in day conditions.

Stepped Frequency Continuous Wave (SFCW) : Alternatively, SFMCW can be used to implement a ToF camera. SFMCW is quite similar to the square wave FMCW, and however, it is achieved by a few successive steps of frequencies. Essentially, the modulation frequency is kept constant, and but, whenever a return signal is detected the frequency is incremented. Therefore, from the step of the frequency, the time of arrival can be measured. This is an attempt on improving the range of the square wave FMCW LiDAR, with resolving the ambiguity problem by introducing multiple frequency steps. The frequency modulation at i th pixel can be formulated as below, where w_i is the frequency at the i th pixel, w_0 initial frequency, x is the distance measured, and the Δw change in the frequency [45].

$$w_i = w_0 + \Delta w \quad (2.12)$$

The measured signal at pixel i can be determined as below.

$$f_i = m \cos(\phi_i) + n \quad (2.13)$$

Where the phase of reflected signal can be expressed as below.

$$\phi_i = \phi_{w_i} = \frac{2w_i x}{C} \quad (2.14)$$

Illustration of the SFCW LiDAR's waveforms can be seen in Figure 2.5. An example of SFMCW LiDARs can be the one described in [51]. In this implementation high

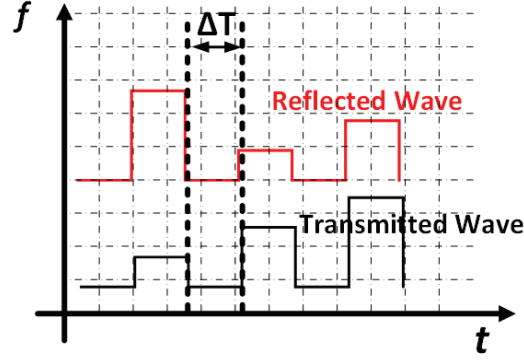


Figure 2.5: Illustration of an SFMCW LiDAR

coherent SFMCW LiDAR was implemented by using frequency stepped pulse trains for the wind velocity measurement. This measurement achieved $1.5 \mu\text{m}$ measurement ranges using light waves synthesized by a frequency sweeper.

2.2.1.2 Pulse LiDAR

Pulse LiDAR uses time quantisation modules to implement range finders. TDCs are commonly used in this process due to their precise time measuring capabilities. Upon the receiving of pulses by the detector, timestamps (time tags) for the events are generated and their timing information is utilised for measuring the time between events. The relationship between the time and distance in a pulse LiDAR can be represented as in equation below, where T_R is the round-trip time of flight for the light pulse, R is the distance between the transmitter and the target and the C is the speed of light. Since the T_R is a round trip for measured time is needed to be divided by 2 for the real distance between the receiver and transmitter [4].

$$T_R = 2R/C \quad (2.15)$$

A diagram for the illustration of a Pulse LiDAR measurement can be seen in Figure 2.6. In Figure 2.6, the pulse laser generates a pulse that travels to the target. When the pulse is created, the START signal is asserted for the time quantisation. When the photon is reflected from the target, it is detected by the photon detector, and the STOP event is generated. Thus, the quantised time difference between the START and STOP is calculated to find the distance [4].

Two types of Pulse LiDAR can be listed; the Single STOP LiDAR and the Multi-STOP LiDAR. These methods differ between the number of STOP signals used to quantise the time differences, and this practice affects the reliability of the operation

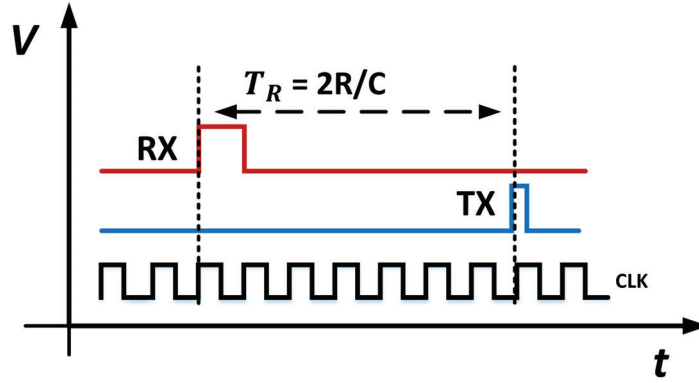


Figure 2.6: *Illustration of a Pulse LiDAR measurement*

while increasing the complexity of the measurement. These methods will be discussed in the following sections [4]

2.2.1.3 Single-Stop LiDAR

Single Stop LiDAR uses the time-of-flight of photons, where the time difference between the transmission and detection of the pulse is used for measuring the distance. Typically, hundred thousands of pulses are sent to the target for this purpose [52]. The time of transmission is determined as the START, and the detection of the signal is defined as the STOP event. Essentially, the time between one START and one STOP is interested in these LiDAR schemes. TDCs are widely used for the time quantisation between the START and STOP signals in pulse LiDARs. The TDC module converts received input pulses into digital timestamps with respect to its system clock. START and STOP signals are quantised in a time-sequenced manner with respect to the same reference point, and hence, subtracting the timestamps of these signals gives the absolute time between these two events [4].

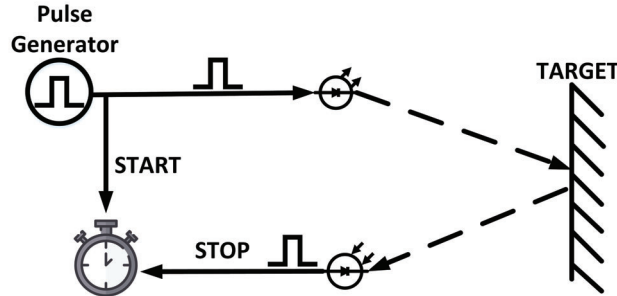


Figure 2.7: *An example of single-stop LiDAR method*

However, since there are many uncorrelated events make the receiver click such

as the background light, dark counts, weak returns and reflections etc., a single measurement between the START and STOP usually will not be enough. Therefore, the pulse LiDAR rapidly sends many light pulses to the target. Received returns are used for forming a histogram from the peak of the time differences. This peak measured from the histogram is primarily used for the value of the measured distance [4].

An explanatory diagram for a single-stop LiDAR architecture can be seen in Figure 2.8.

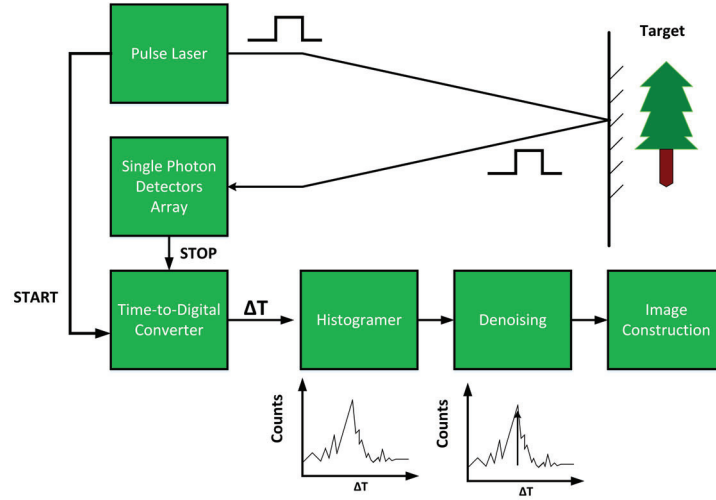


Figure 2.8: *Illustration of a single-stop LiDAR architecture*

In the literature, there are some examples of high precision pulse LiDAR schemes. One of the case to pulse LiDAR systems which use a TDC can be [53], where an ADC based TDC implemented on an FPGA used for a LiDAR managed to measure distances with 40 mm precision. Another example, a CMOS based TDC LiDAR system is described in [54], where 10 ps SSP TDC was used for the LiDAR, which was corresponding to ± 3 mm measurement precision, and also, in [55] by using an FPGA based TDC with 181 ps SSP LiDAR system was implemented successfully.

2.2.1.4 Multi-Stop LiDAR Technique

An alternative to a traditional single stop LiDAR to improve the reliability of the measurement, the Multi-STOP time differencing has been developed [56, 57]. The main difference between this method and the single-stop LiDAR is measurement does not have to stop with the stop signal, and it records the time differences between the START and every STOP signal occurs before the next START. This implementation reduces the errors affecting the LiDAR in environments such as foggy weather, where returns are weak, and many uncorrelated photons are detected from reflections. The

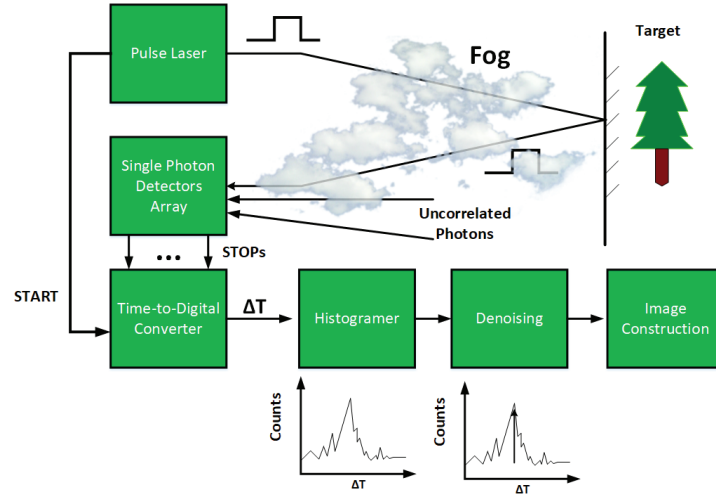


Figure 2.9: An example Multi-STOP LiDAR architecture

reason it improves the precision is if there is an uncorrelated STOP, this will not interrupt the correlated STOP from being detected since until the next START signal all the STOPs are recorded [56, 57].

As it was described in [56], this method can be implemented by using Multi-Stop TDCs, which measure time-differences by generating time tags for received trigger signals. At least two-channels of the TDC is required, but it can be scaled up to as many as needed channel numbers. One of these channels is chosen to be the reference channel, and thus, it generates time tags for the START signals. The reference signal is directly connected to the transmitter. Other channels are connected to detectors and make time tags for triggers detected as STOP signals. This effectively measures the delta times between the START and n number of the STOP signals. The n -STOP delta time measurement can be formulated as in Equation 2.16.

$$\Delta T_{(n.STOP-START)} = \begin{cases} T_{STOP_1} - T_{START} \\ T_{STOP_2} - T_{START} \\ T_{STOP_3} - T_{START} \\ \vdots \\ T_{STOP_n} - T_{START} \end{cases} \quad (2.16)$$

After tags are generated a module named the multi-channel tag correlator is used for calculating the time differences between time tags from different channels. This module continuously subtracts the STOP channel's tags from the START channel's

and outputs the delta times between them. Once the time differences are found, they are output, and a histogram is formed based on these differences. As a result of the measurements, correlated delta times form a Gaussian peak. Amongst all-time differences in the histogram, this peak represents the actual time difference between the START and STOP.

The explanatory diagram for this method can be found in Figure 2.9. For the performance improvements achieved with this method [57] can be referred.

2.2.2 LiDAR Summary

To sum up, LiDAR is a methodology used for measuring ranges by using the time difference between the event of laser transmission and photon detection by the detector. LiDAR techniques typically aim to provide a precise timing measurement of photon's arrival time through the time correlation with respect to a common reference event. These techniques can also be used for the time-correlation in other applications such as Quantum Photonics. There are commonly two types of LiDAR approaches, which are Modulation and Pulse LiDARs. The advantages of the modulation LiDAR are it is less affected by the interference with the other signals, it allows measurement of moving objects' velocities due to the Doppler effect and quicker processing times since the histogramming is not necessary for measurements. Therefore, they are preferable for applications such as automotive. However, these advantages are also observed as complexity in both laser and the data processing end [58]. One of the reasons for the complexity is the requirement for coherent detectors, which mixes the received weak signal with the constant source for the signal reconstruction. Then, a low pass filter is applied to the summed frequency, and the baseband signal frequency is obtained [59]. Also, they are promising to provide longer distances with less power, which implies improved safety features for implementations. Although, the complexity of the required signal processing, this type of LiDAR offers more information about the target, and but, for applications, where targets are stationary, this type of LiDAR is not necessary. Thus, the information it provides not needed, and the complexity of the data processing can be spared for areas such as improving the timing precision and data-rate.

On the other hand, a pulse LiDAR is more straightforward in terms of implementation, and precise results can be achieved without sophisticated signal processing techniques. A pulse laser sends a burst of pulses to a target and detectors record the reflected pulses' triggers. Then, the correlation of arrival times is measured from a histogram. This is done by splitting the power output of the pulse laser into smaller chunks and spreading their projection over time. In most of the quantum photonics experiments, the measurement distances are short, targets are stationary, interference and eye safety are less of a concern compared to the outdoor applications. Thus, the

pulse LiDAR can be effectively used for the precise measurement of the time of arriving photons [4].

2.3 Quantum Information

The discovery of quantum mechanics stretches back to the invention of cathode rays by Michael Faraday in 1838 [60]. Later by Julius Plucker (1858), magnets' effect on the electrical discharge in rarefied glasses was shown by using cathode rays, which led to the discovery of electrons. Plucker showed that when the electric current was flown through a vacuum tube, a free movement of electrons was observed by the emission of green light as the energy levels of electrons are changed [61].

Upon the Ultra-violet Catastrophe in the 19th century, laws of classical Newtonian energy conversation laws of physics failed to explain the blackbody radiation. As in classical physics suggests an infinite spectrum of ultra-violet light is emitted as the energy increases, and however, the observed data had shown an unexpected behaviour of light [61].

Later in the early 20th century, this mathematical error was fixed by Max Planck by inventing the Planck constant from the Weiss approximation, and his invention led to the understanding of how light rays travel as in bundles or 'quantas' and how they can transfer energy from one quanta to another. Thus, the energy levels in an atom's orbit are explained by the Planck's Constant. The n th level of energy around an atom is formulated as below.

$$\Delta E = nhf \tag{2.17}$$

This discovery led to a departure from classical mechanics into new quantum mechanics. Planck's constant has become one of the fundamental constants in nature. Later, the photoelectric effect of light explained by Einstein, where the light's property is not defined as in a classical wave theory instead it preserves energy as bundles, and these bundles are later called photons [61].

Photons are massless fundamental particles in a quantum of electromagnetic radiation in light and radio waves. Photons do not possess electrical charge, and however, they do have properties such as polarisation, position, momentum and magnetic spins. In addition, they travel with the speed of light in a vacuum. Photons are found in a quantum, where they are in random states, and their properties can be only measurable once at a time due to the Heisenberg Uncertainty principle [62]. Upon a measurement, photons stay in the state of the measurement and their quantum property cannot be irreversible to the original state.

The phenomenon called the quantum entanglement occurs when photon pairs are generated in a way that their states cannot be described separately from each

other. Hence, the properties of the photon such as polarisation, momentum, spins and positions are correlated among pairs. Measuring a single-photon in a pair results in photons to freeze their quantum properties at the state they are measured. The measurement of one of the pair results in both pair to fix their properties and the other pair to be updated related to the measured one's property. Therefore, measuring one of the pair reveals information about the other one and allows entangled photons to be in two states at the same time.

With recent developments in quantum physics were led the classical information theory to evolve into quantum information theory. Unlike in the classical information theory, where the information is encoded on a single bit with only being able to hold one of the two states of '1' or '0', Qubits are in a coherent superposition of two different states at the same time. For instance, in a two-state quantum system, two different polarisation states of photons can be encoded into one Qubit, and these states keep the quantum system in various states simultaneously [63]. As a result quantum computers can compute exponentially faster than a classical computers. For instance, if quantum computer has 500 qubits, this indicates 2^{500} quantum states where each state operates as a single classical computer with a 500-bit string of '1's and '0's. This kind of computer would yield to a computation power of 10^{150} classical processors [64].

2.3.1 Quantum Bits

Quantum mechanics states that a system can exist in a superposition of states. These quantum states are expressed by using the bra-ket notation (Dirac), which is used for representing vectorial combinations of $|0\rangle$ $|1\rangle$. The state Qubit in $|\phi\rangle$ can be expressed with probability amplitudes of the system being in $|0\rangle$ and $|1\rangle$, where probability amplitudes are used for expressing the probability of complex numbers in a vector space for quantum systems. When probability functions are represented by α and β the state of Qubit can be expressed as below [63].

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (2.18)$$

Since both α and β are complex numbers, the probability of $|0\rangle$ is $|\alpha|^2$ and the $|1\rangle$ is $|\beta|^2$. Hence,

$$1 = |\alpha|^2 + |\beta|^2 \quad (2.19)$$

This ensures α and β can be 1,0 but not a value in between. Also, the superposition of states defines that it is not possible to know which states are forming the output. However, the phase between the probability amplitudes of α and β is related to a phenomenon called the quantum interference [63].

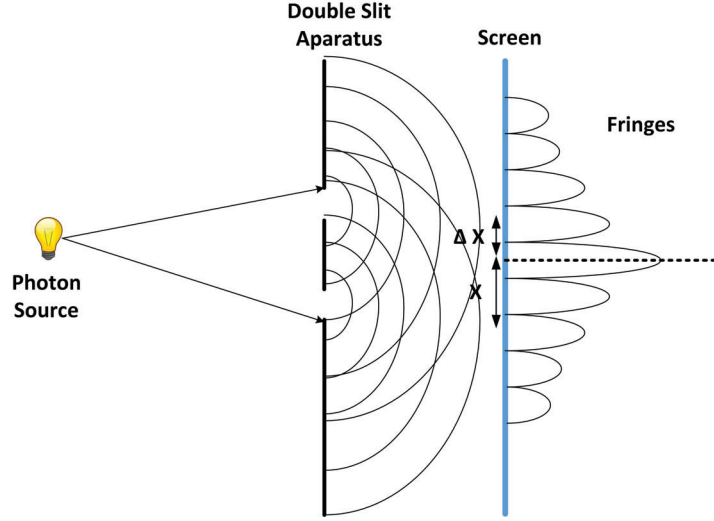


Figure 2.10: *Illustration of Double Slit Experiment*

2.3.2 Quantum Interference

The quantum interference is a phenomenon which occurs when photons or particles are projected, and the interference is observed at their projection. This was first observed in the famous double-slit experiment by Thomas Young in 1801. As the wave theory suggests, light waves consist of stream photons, and thus, their projections on the target should cause wave-like interference patterns. However, when the photon source's intensity is lowered to the point, where only a single-photon can be emitted at a time, after waiting long enough, the same wave-like interference pattern would not be expected to be observed. Yet, exactly the same interference pattern occurs as a result of the experiment. Thus, the interaction between photons does not cause interference, but the interference of photon with itself. Therefore, there is a probability of photons hitting certain areas higher whilst, other areas lesser. As the probability increases, the fringe appeared to be brighter. This is explained by the quantum superposition, where photons can be in two different states at the same time. So, it passes both slits simultaneously and interfere with itself and explains the Double Slit Experiment from a quantum mechanics point of view [65].

Two regions where photons can hit can be defined as x and $x + \Delta x$, where Δx is smaller than x . Since each photon preserves the same amount of energy, the number of photons illuminate the surface in a given time interval is expressed proportionally by the intensity of light and area of the illumination surface [65].

$$P_x(x) \approx \lim_{K \rightarrow \infty} \left[\frac{k(x)}{K} \right] \propto I(t)\Delta x \quad (2.20)$$

In Equation 2.20, $P_x(x)$ is the probability of a photon to hit the surface between x

and $x + \Delta x$ can be seen, where $I(t)$ intensity of light for the time t . The probability of photon to hit on a region can be either 1 or 0, and this probability is proportional with the width the Δx ($P_x \propto \Delta t$). Hence, the probability density, $P(x)$ for the unit width dx can be defined as $P_x(x) = P(x)dx$, and since, the $P_x \propto \Delta t$ then $P(x) \propto I(x)$. Due to the Born's rule [65], which states the density of function's proportion to the wave function as $I(y) \propto |\psi(x, t)|^2$, the probability function of a single photon hitting to the point x can be expressed as below.

$$P(x) \propto |\psi(x, t)|^2. \quad (2.21)$$

An expression of a single photon interference with itself can be expressed as a probability as in Equation 2.3.2, and however, the exact position, where photons will hit cannot be known and can be only expressed as a probability in quantum mechanics [65].

2.3.3 Hong-ou-Mandel Effect

Previously, the interference of a single photon with itself due to the superposition principle was discussed. Also, to the single-photon interference, two identical single-photons interference is an essential concept in quantum photonics. This type of interference is known as a Hong-ou-Mandel (HOM-dip) Effect as it was first described in [66]. The phenomena of HOM-dip is observed when two identical single-photons enter a 50:50 beam splitter, and a cancelling effect of photons scrutinised at the output. The input of a single photon to a 50:50 beam splitter causes an equal chance of being reflected or transmitted, and thus, there are four possibilities. These are, they are both reflected or transmitted; either one of them is reflected while the other is transmitted. These situations can be seen in Figure 2.11.

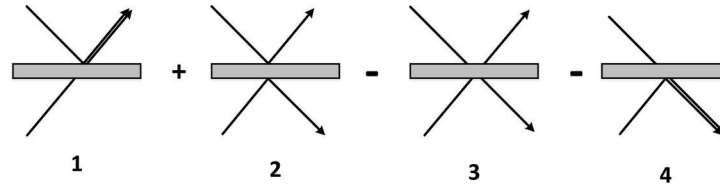


Figure 2.11: *Illustration of a HOM Dip, where 4 different outputs of 50:50 beam splitter was explained*

When situations 2 and 3 occur in Figure 2.11, the cancellation of both photons is observed as a dip at the coincidence rates of photons upon detection, and thus, they become indistinguishable. The concept of being identical depends on the sensitivity of an experiment which can be the phase, polarisation or spin of photons. Therefore, this interference could be used to correlate the arrival times of two entangled photons to

the beam-splitter due to their identical arrival times and this property can be modelled mathematically as below [67].

$$|\psi_{in}\rangle_{ab} = \hat{a}_n^\dagger \hat{b}_m^\dagger |0\rangle_{ab} = |1;n\rangle_a |1;m\rangle_b, \quad (2.22)$$

In Equation 2.22, \hat{a}_n^\dagger and \hat{b}_m^\dagger are bosonic creation operators for beam splitter modes of photons a and b. The sensed properties of photons a and b by the detectors are n and m, where they can represent different polarisations, spins or phases etc. These are the properties used to measure the distinguishability of photons. Modes of a beam splitter can be seen in Figure 2.12 [67].

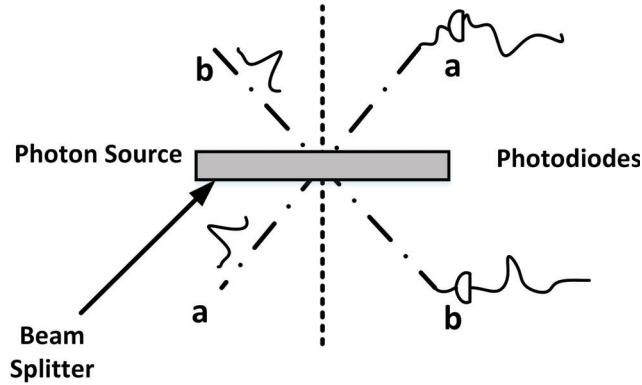


Figure 2.12: Illustration of the beam splitter modes

When a 50:50 beam splitter is modelled mathematically, it is essentially an application of unitary matrix to the input photons, where the reflectivity of a splitter is $1/2$. The beam splitter's transformation can be modelled as below [67].

$$|\psi_{out}\rangle_{ab} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \hat{a}_n^\dagger \\ \hat{b}_m^\dagger \end{bmatrix} \quad (2.23)$$

Therefore, it can be expanded as in Equation:2.24 [67].

$$|\psi_{out}\rangle_{ab} = \frac{1}{2}(\hat{a}_n^\dagger \hat{a}_m^\dagger + \hat{a}_n^\dagger \hat{b}_m^\dagger - \hat{a}_m^\dagger \hat{b}_n^\dagger - \hat{b}_n^\dagger \hat{b}_m^\dagger)|0\rangle_{ab} \quad (2.24)$$

If photons' polarisations are chosen to be the distinguishing sensitivity, then the Equation 2.24 can be rearranged as below, where the horizontal polarisation is H and the vertical polarisation is V ($n = V$, $m = H$).

$$\begin{aligned} |\psi_{out}\rangle_{ab} &= \frac{1}{2}(\hat{a}_V^\dagger \hat{a}_H^\dagger + \hat{a}_V^\dagger \hat{b}_H^\dagger - \hat{a}_H^\dagger \hat{b}_V^\dagger - \hat{b}_V^\dagger \hat{b}_H^\dagger)|0\rangle_{ab} \\ &= \frac{1}{2}(|1;H\rangle_a |1;V\rangle_a + |1;V\rangle_a |1;H\rangle_b - |1;V\rangle_b |1;H\rangle_a + |1;H\rangle_b |1;V\rangle_b). \end{aligned} \quad (2.25)$$

CHAPTER 2. RESEARCH REVIEW

In Equation 2.25, the first term represents the case when both photons leave the splitter from mode a, and the last term represents when both photons go from mode b. The second and the third terms represent when photons leave the splitter from both a and b modes. Thus, there is a 1/2 probability of coincidences due to 2 out of 4 cases involving photons leaving the splitter in both modes with orthogonal polarisations, and this makes them distinguishable. However, it should be noted that it is only valid when $n \neq m$ [67].

When $n = m = V$, Equation 2.25 can be rearranged as below.

$$|\psi_{out}\rangle_{ab} = \frac{1}{2}(\hat{a}_V^\dagger \hat{a}_V^\dagger + \hat{a}_V^\dagger \hat{b}_V^\dagger - \hat{a}_V^\dagger \hat{b}_V^\dagger - \hat{b}_V^\dagger \hat{b}_V^\dagger)|0\rangle_{ab} \quad (2.26)$$

Hence,

$$\begin{aligned} |\psi_{out}\rangle_{ab} &= \frac{1}{2}(\hat{a}_V^\dagger \hat{a}_V^\dagger - \hat{b}_V^\dagger \hat{b}_V^\dagger)|0\rangle_{ab} \\ &= \frac{1}{\sqrt{2}}(|2;V\rangle_a - |2;V\rangle_b). \end{aligned} \quad (2.27)$$

As it can be seen in Equation 2.3.3, the middle terms of the equation cancel each other out since they have opposite signs. Thus, when identical photons leave the splitter they extinguish each other as this is observed as a dip in the coincidence rates and makes them indistinguishable. The reverse of this property (Rev-HOM dip) is also used for the expressing the two-photon interference between two identical photons. The reverse fringe of the HOM-dip effect is measured by coincidence counter instruments for this type of interference [67].

Some usage of this effect can be expressing the fidelity of the quantum chips used for generating coherent-states, where the degree of coherence is measured by using the HOM-Dip effect [68] and measuring the purity of single-photon sources from the interference visibility parameter for quantum systems since the visibility of photons is directly related with the HOM-Dip effect [67]. The details of these processes are not in the scope of this thesis. However, the measurement of Rev-HOM dip interference by using a coincidence counter will be discussed in Chapter 7.

2.3.4 Quantum Communication

One of the applications of quantum information theory is quantum communication. Today, lots of confidential data is transferred across optical channels all around the world. Typically, the data is mapped on digital bits '1' and '0', and classical cryptography protocols are used to encrypt the data to transfer them securely. However, quantum communication systems benefit from the quantum mechanics proprieties to map the

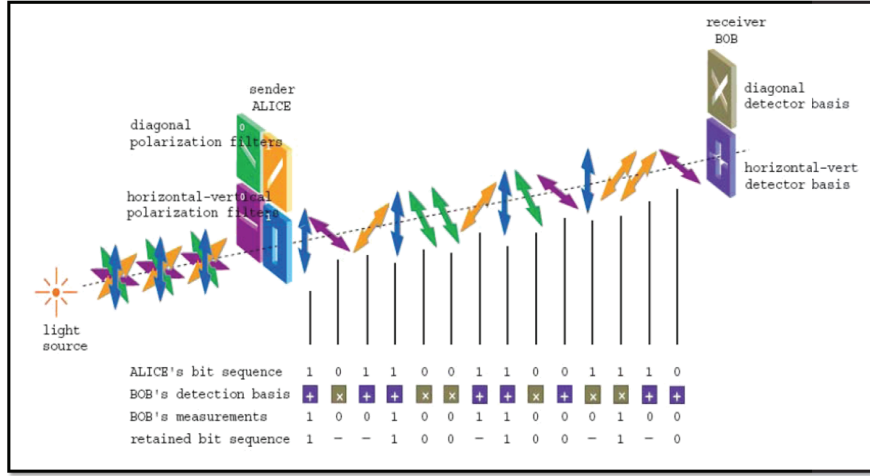


Figure 2.13: Demonstration of BB84 QKD scheme [5]

information to qubits, and this makes the communication extremely sensitive to eavesdropping. Thus, in case of eavesdropping, it results in a collapse of the coherent quantum states into one due to the no-cloning theorem [69].

Probably, the most well-known quantum cryptography protocol is BB84 Quantum Key Distribution (QKD) which maps the secure key on different polarisation states to be exchanged between parties. Typically, in cryptography, these two communicating parties are referred to as Alice and Bob, and the party that eavesdropping is named as Eve. The BB84 protocol operates with 2 different polarisation basis; rectilinear and diagonal, and 4 different bit encoding depending on the polarisation modes of photons; 0° , 45° , 90° and 135° [70].

The protocol starts with Alice's communication with Bob over a quantum channel. Alice randomly generates a string of bits. These bits are randomly encoded on a different polarisation basis (i.e. rectilinear or diagonal). Then, Bob decodes each received bit using randomly chosen polarisation basis. Therefore, if they are both chosen the correct basis, both Alice and Bob will have the same bit [70].

After the transfer of the key is finished, over a public channel, Bob notifies Alice about the order of the polarisation basis he used to decode the bits. Alice replies whether he used correct polarisation basis for each bit. Later, both Alice and Bob discard incorrectly measured bits, so they obtain the same key for the data encryption. Discard of incorrect bits is named as the Key Sifting [70]. After the Key Sifting, randomly chosen bit words of the data is compared between Alice and Bob to determine the amount of eavesdropping and the consistency of the data. Demonstration of BBQ84 protocol with Key Sifting can be seen in Figure 2.13.

The security of this scheme is due to the Heisenberg Uncertainty principle, and if eavesdropping is encountered during the mid-transfer, it will be noticeable. Since

CHAPTER 2. RESEARCH REVIEW

Eve has to choose random polarisation basis to measure the polarisation, the wrongly measured bits will result in a permanent loss of the data. Thus, Bob will read a random bit, and if Both Bob's and Eve's chances of correctly choosing the polarisation basis is 50%, the probability of Bob's accurately measuring each bit drops to 25%. When a very long data streams with n bits is chosen to be transferred, there will be a $(\frac{3}{4})^n$ chance of detecting the Eve [70].

Alternatively, a BB84 variation uses entangled photons can be used to implement a secure key exchange as it described by Arthur Eckert [71]. This protocol suggests the use of entangled photon pairs to implement a QKD protocol. This protocol is based on performing an entanglement check done on measured photons, and due to the Bell's theorem, the data received by both parties should be a binary complement of each other. However, when the same bit is captured by both Alice and Bob, this will indicate an eavesdropping [70, 71].

In some QKD systems, continuous-variable quantum cryptography is used [72]. By generating coherent states of light, non-orthogonal phases of the light (i.e. Binary Phase Shift Keying (BPSK)) and the amplitude modulation of light can be used to encode the data [73]. Thus, when the phase is used $\Delta\phi_{Alice} - \Delta\phi_{Bob} = \Delta\phi$ and hence $\Delta\phi = 0$ it is interpreted as a bit '0' [72], when $\Delta\phi = 180^\circ$ as a bit '1'. When the amplitude modulation is used, Alice sets n bits to the amplitude range and depending on the measured amplitude, the bits are decided by Bob. After, the transmission via a public channel, Bob shares whether he used an amplitude or phase measurement for the data and the previously mentioned Key Sifting is applied [73].

Coherent states in quantum mechanics represent quantum mechanical states of light which are in the mode of the quantum harmonic oscillator [74]. For instance, in a QKD system, Alice generates N number of coherent states of light and the N states of the encoding scheme is named as the constellation $C(z, N)$. One of the coherent states is transferred to Bob through a thermal noise channel [6]. The interference of Eve is modelled as a beam splitter. In this scheme from the loss in the thermal noise channel, the amount of eavesdropping can be estimated. Illustration of this system can be seen in Figure 2.14, where A is a mode in four coherent states of a constellation with a radius of r , and sent to Bob through Eve's thermal noise channel. A beam splitter with a reflectivity of t is used to model the thermal channel loss. E and e are modes of Eve which are used to attack Alice's Mode. E' and B are attenuated coherent states, and B is also the detection mode of the Bob [6].

Another important tool used in quantum communication is Photon Number Resolving Detector (PNRD). PNRD is used for measuring the number of photons present in optical pulses [75]. A precise PNRD should be able to count photons with very short pulses, and they are commonly used in many different quantum information and quantum communication applications such as QKD systems. Their utilisation in quantum systems is to monitor sources. For example, they can be used for characterising the

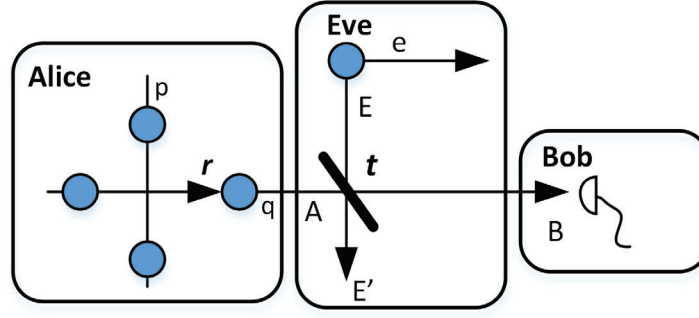


Figure 2.14: *Demonstration of continuous variable quantum cryptography [6]*

untrusted sources in QKD systems [76] and characterising light sources in quantum systems [77]. The details of these operations are not in the scope of this thesis, but an example of how an FPGA based coincidence counter is used to implement a PNRD can be found in Chapter 7 also in [3].

2.4 Technologies Available for Timing Measurement Tool

Precise time measurement tools have been implemented by using different technologies over the years. Depending on the needs of applications different technologies can be desirable due to their distinct advantages and disadvantages. When choosing a technology to implement a timing tool, typically, performance, precision, area utilisation, power and cost are considered. In the literature, the majority of the TDC architectures uses technologies such as FPGAs, ASICs and analogue circuits. These methods are preferable over synchronous Integrated Circuits (ICs) such as CPUs, GPUs and DSPs because the system's clock does not limit their precision. However, due to the high dead-times of analogue circuits such as Time-to-Analogue Converters (TACs) [1], FPGA and ASIC technologies are more desirable for a high count-rate timing instrument due to their capabilities of accommodate fast asynchronous and concurrent logic designs. These instruments are proven to be measure timings with more precision than system clock speeds [1]. Typically, ASICs are prohibitively expensive compared to FPGAs as the break-even point between ASICs and FPGAs is around 100 thousand units per manufacturing run [78]. Also, FPGAs provide more flexibility in applications due to their re-programmability in the field. FPGAs, Application Specific Integrated Circuits (ASICs) and Analogue Circuits which are utilised for implementing timing circuits will be discussed in this section.

2.4.1 Field Programmable Gate Arrays (FPGAs)

FPGAs can be defined as semiconductor circuits which accommodate configurable logic elements and switch matrices in order to be re-programmable based on the required changes in applications [79]. FPGAs' roles are not predefined, and they can be easily adapted to a change in the application's environment. They are typically programmed by using VHSIC Hardware Description Language (VHDL) (Very High-Speed Integrated Circuit (VHSIC)) or Verilog and their variants. The structure of FPGA consists of reconfigurable logic blocks, component switching elements, hard-core components such as Digital Signal Processor (DSP) blocks, accumulators, multipliers, RAM Blocks, clock managers, carry chains, FPGA interconnects and soft-core embedded processors. An example of an FPGA fabric can be seen in Figure 2.15.

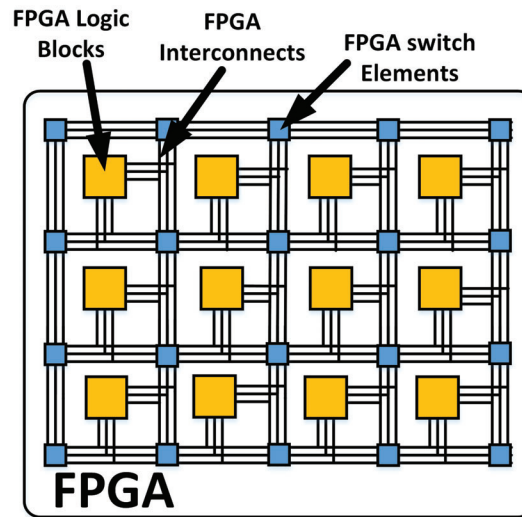


Figure 2.15: *Illustration of a FPGA Fabric*

With the recent developments in FPGA technology, the fabric started to reach clock speeds of 500 MHz, and their fast circuit structure provides superior performance compared to other sequential operating integrated circuit technologies [80]. Areas, where FPGAs are used, include aerospace, defence, ASIC Prototypes, audio processing, automotive, consumer electronics, data centres, security, medicine, communications and photonics [81]. The main performance advantage of FPGA technology comes from its strictly concurrent nature, which provides very fast operational speeds due to its logic-based architecture. Also, the usage of asynchronous logic elements implementations can achieve performance which is not limited by the system's clock period, and it is especially desirable for the fine time measurement instrument.

One of the most significant drawbacks of FPGA technology is the power and area utilisation. The non-utilised logic blocks cause these issues in FPGA designs which

2.4. TECHNOLOGIES AVAILABLE FOR TIMING MEASUREMENT TOOL

consume power unnecessarily and result in additional power requirements to meet the timing constraints. According to research done in this area by Kuon I. and Rose, J [82], the implementation of a Static Random Access Memory (SRAM) on both 90-nm FPGA and 90-nm ASIC device are compared by their performance. This research showed the average silicon area required to implement lookup tables in FPGA is around 35 times more than its ASIC counterpart. Also, the effect of hard operational cores such as DSP multipliers and accumulators usage on the logic utilisation was observed to be reducing the logic utilisation up to 18 times. When the critical-path delays are compared, FPGAs typically have three to four times larger, and the total dynamic power consumption of an FPGA is usually 14 times greater than an ASIC. Also, having a larger silicon area than needed results in power fluctuations, temperature fluctuations and non-linear propagation delays between FPGA elements.

Important FPGA manufacturers in the market are Intel (ex-Altera), Xilinx, Lattice and Microsemi (ex-Actel). Xilinx provides FPGA solutions ranging from 45nm to 16 nm and the oldest brand in the market. Their Spartan, Artix, Virtex, Kintex and UltraScale FPGAs are popular among FPGA users. On the other hand, Intel recently who bought Altera providing high-performance solutions which integrate System on a Chip (SoC) in their 28nm-10nm FPGAs devices. Some of their popular models include Agilex, Startix, Arria and Cyclone. Other smaller-scaled manufactures like Lattice and Actel aims to provide more budget solution in FPGAs and Complex Programmable Logic Devices (CPLDs) to less computational power required implementations such as sensor interfacing. It is estimated that Xilinx holds 50 % of the market, Intel % 37 , Lattice % 10 and others %3 [83].

2.4.2 Application-Specific Integrated Circuits (ASICs)

ASIC technology is very similar to FPGA; the main difference between FPGAs and the ASICs is ASICs' silicon is customised for the needs of the applications. Therefore, large empty blocks of silicon, large power consumptions, substantial propagation delays and problematic timing constraints are lesser issues. They typically achieve better timing performance than FPGAs due to their optimised nature. However, this is reflected in their costs, especially for the low unit manufacturing, FPGAs are more cost-efficient for small to an average number of units. If a vast amount of chips are needed to be shipped, then ASICs can be more cost-efficient. Hence, roughly around 100 k units of ASIC become more cost-efficient than the FPGAs at the same number of units [78].

In the price point, FPGAs cost much less than ASICs since they are widely mass manufactured. Typically, the price of a modern FPGA chip would cost between ≈ 20 dollars to 40,000 dollars, and by using ≈ 100 -160 dollars an FPGA high precise timing instruments could be implemented [84]. Thus, FPGAs can be considered as plausible options for timing measurement tools.

The types of ASICs are typically separated into two different categories which are Fully-Customised and Semi-Customised ASICs [85]. Fully-customised ASICs have all the circuit components and interconnects customised for applications. Therefore, they achieve the best performance with minimum logic utilisation and power. However, they are less flexible, costly and longer to manufacture. Also, ASICs are not re-configurable like FPGAs. Illustration of ASIC types can be seen in Figure 2.16

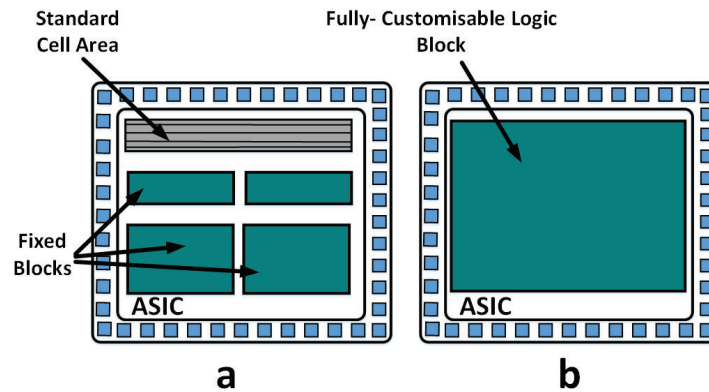


Figure 2.16: *Illustration of ASIC types*

Semi-Customised ASICs are ASIC variations, where some level of customisation is introduced, but not fully customisable. They are typically divided into two categories which are Standard Cell Based ASICs and Gate Array-based ASICs. Standard cell area based ASICs usually has standard cell libraries which hold the implementations of logic gates, multiplexers and flip-flops. These are designed by designers and fixed blocks which utilises these standard cell areas. Gate Array-based ASICs have fixed logic blocks, and but, based on the application's needs, the designer can modify the interconnects. Semi-Customised ASICs are less costly compared to Full-Customised ones, and however, they are performing somewhere between FPGAs and Full-Customised ASICs.

2.4.3 Analogue Circuits

A final approach to implement a timing instrument can be using analogue circuits. Analogue Circuits can be implemented Printed Circuit Board (PCB) and Complementary Metal-oxide-semiconductor (CMOS) technology to design an instrument. Integrated Circuits (ICs) such as Analogue-to-Digital Converters (ADCs), counters and other circuit components can be counted in this area. The precision of these implementations are limited by the quality of the components used for these implementations [86, 87].

Although, an analogue circuit can achieve a very high precision in terms of the time measurement, the main drawback of an analogue circuit is the dead-time between two

2.4. TECHNOLOGIES AVAILABLE FOR TIMING MEASUREMENT TOOL

measurements. This is mainly due to the need of charging and discharging processes between measurements. Typical implementation examples to this type of technology could be Time Expansion (TE) and Time-to-Analogue Converter (TAC) methods. The details of these methods will be discussed in the next section.

2.5 Time-to-Digital Converters (TDCs)

Time-to-Digital Converters are precise time measurement instruments commonly utilised in many different aspects of science and engineering. Their applications become widely popular as the popularity of the ToF applications increase. TDCs can be found in applications such as autonomous vehicles as the part of the LiDAR systems [88], ToF cameras in medical imaging such as PET cameras [31], part of the clock recovery in QKD [89], and also, as non-ToF examples such as phase detectors in Phased-Locked Loops (PLLs). The concept of TDC is closely related to ADCs; even some implementations use ADCs to implement TDCs. Both of these instruments are used for converting input signals into output signals in another domain. ADCs operate between analogue-to-digital domain by converting analogue signals to digitised values, whilst TDCs convert time to digitised values. Since TDCs and ADCs are closely related, concepts such as resolution, dead-time, linearity and errors associated with the conversion process are shared between them. These concepts will be discussed in the further sections of this chapter.

Traditionally, TDCs are commonly used for quantising the time interval defined by the START and STOP signals. In the literature, TDC implementations are divided into two main categories; coarse and fine TDCs. Coarse TDCs typically use counters or variations of counters to achieve time quantisation whilst; fine TDCs aim to provide precision which is not limited by the system clock (T_{clk}). Typically, a coarse counter TDC start counting with the START signal and outputting the counter value with the STOP signal. However, for applications require higher precision, fine TDC variations are commonly used.

Fine TDCs aim to provide sub-clock period measurements. However, fine TDCs have limited measurement ranges because they are specialised in measuring short ranges such as sub-nano seconds [90] and even sub-pico seconds[91]. Thus, coarse counter TDCs are preferred for measuring longer distances while fine TDCs are for shorter time intervals.

TDC can be implemented by using various fabrics and circuits. Most of the TDC implementation available today use technologies such as FPGAs, ASICs and Analogue Circuits. Based on the smallest distance can be measured by the TDC, the resolution of the TDC is defined. The resolution represents the LSB of the timestamp, and it should not be confused with the precision. Hence, the precision defines the error margin affecting each measurement.

Further in this section, the available TDC implementation, calibration and linearisation methods and the errors affecting TDCs will be discussed.

2.5.1 Coarse Counter TDC

Probably the most basic TDC implementation available would be a coarse counter TDC. In this method, with an assertion of the START signal, a counter starts counting at every clock edge and outputs the counter value with the assertion of the STOP signal. Hence, the time frame between START and STOP signals are quantised with respect to the system's clock edge. This time quantisation can be expressed like in Equation 2.28, where ΔT is the time frame, T_{START} and T_{STOP} are the START and STOP events respectively [1].

$$\Delta T = T_{START} - T_{STOP} \quad (2.28)$$

A diagram illustrates a coarse counter and its waveforms can be seen in Figure 2.17.

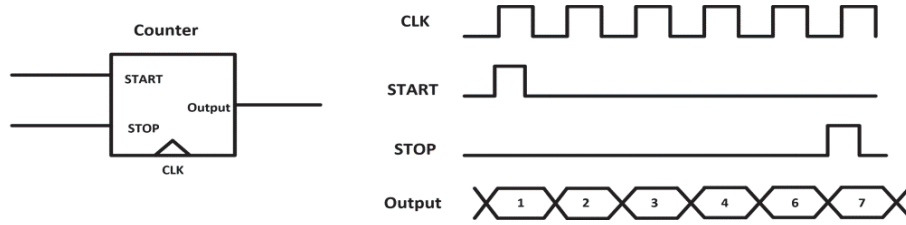


Figure 2.17: *Illustration of a coarse counter TDC and its waveforms*

The advantage of using this scheme would be its simplicity compared to the other TDC implementations. A coarse counter TDC utilises less logic since the only required operation is registering the counter values as the counter increments. Thus, scaling up the number of operational channels could be achieved easily.

The main limitation of this type of TDC is the resolution being limited to the system's clock frequency. Therefore, the measurement of sub-clock period time intervals is not possible to be achieved. The limit of how fast any oscillator can operate within the logic circuit is mainly limited by two factors; the power consumption and speed that logic can work. Increasing the number of bits that represents the tag will require an increase in the clock frequency exponentially. The maximum operable clock frequency will limit this by the logic elements. Also, an increase in the power consumption of a clock will increase since a faster clock will require higher voltage values to meet with the timing constraints. Thus, the voltage that can be supplied will be the limiting factor.

Coarse counters are commonly used in applications, where low resolutions are acceptable, due to their simplicity. In a typical high-resolution TDC architecture, coarse counters are used as a part of the hybrid interpolation schemes. The coarse counters are employed for measurement ranges longer than a clock period while the fine interpolation is used for measuring intervals shorter than a clock period.

CHAPTER 2. RESEARCH REVIEW

Another variation of a coarse counter TDC is called the Multi-Phase Clock (MPC) TDC, which utilises multiple phase-shifted clocks to generate different counters. Each counter counts slightly out of phase from each other with a known phase difference. MPC effectively increases the resolution of the counter, and however, as the number of bits added to the code the number of clocks with different phases required increases exponentially. This method requires a reasonably high area and logic utilisations for high resolutions, and so, not ideal compared to the other methods such as delay lines [92].

The multi-phase counter uses the total number of clock edges like in the coarse time quantisation, and the number of clock periods passed is defined by $n \times T_{clk}$. However, the passed phases ($\Delta\phi$) implements the fine timing for the TDC. $\Delta\phi$ in this implementation refers to the number of completed full cycles before the STOP signal. The time difference of T_{diff} is a combination of these two measured quantities. The equation for the time difference can be derived as below.

$$T_{diff} = n * T_{clk} + \Delta\phi \quad (2.29)$$

An illustration of a multi-phase counter implementation can be seen in Figure 2.18.

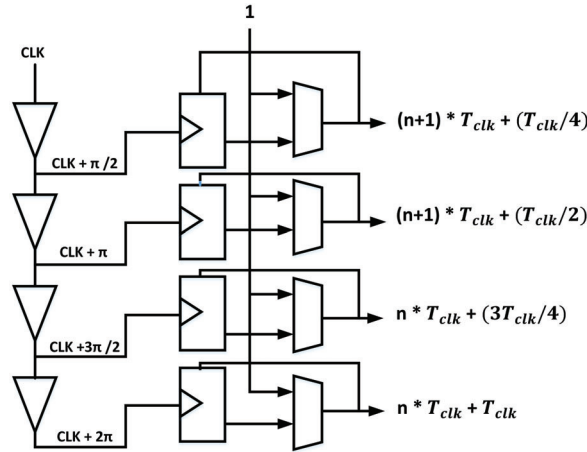


Figure 2.18: *Illustration of Multi-phase counter*

An example of the waveforms of a multi-phase counter can be seen in Figure 2.19. In Figure 2.19, the time difference can be expressed like $T_{diff} = n \times T_{clk} + \Delta p/16$. Hence, the bottom clock is the only clock which could not complete its 2nd cycle thus the total clock phases completed by 8 multi-phase clocks were 15.

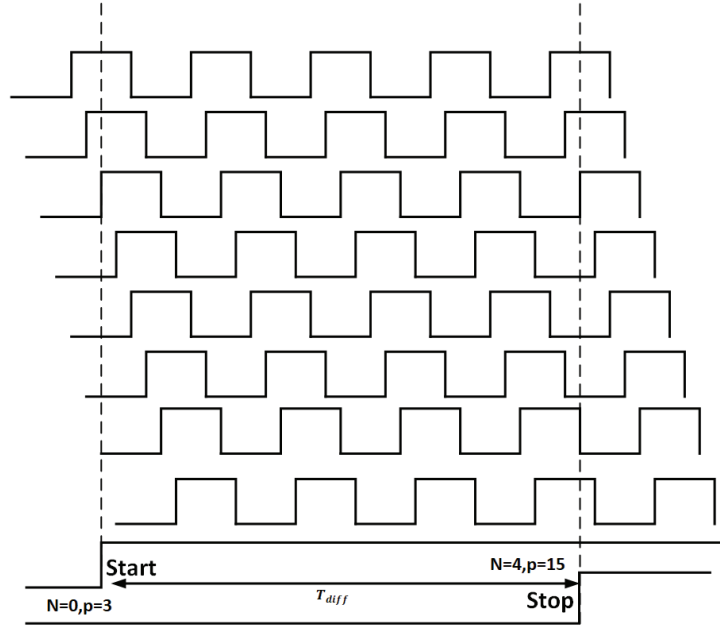


Figure 2.19: *Multi-phase counter waveforms*

2.5.2 Hybrid Interpolation

The hybrid interpolation (Nutt Interpolation) scheme, is a method of combining fine and coarse interpolation schemes, to provide TDC architectures which can count longer ranges with higher precision. The hybrid interpolation operates with by measuring the time intervals greater than the clock period by a coarse counter while a fine time TDC measures the intervals shorter than the clock period. The Nutt interpolation is originally described in [93]. In Equation 2.30, a formulation of Δt measured by the Nutt interpolation can be seen, where n is the required clock cycles (T_{clk}), F_{START} is the fine time for the START and F_{STOP} is the fine time for the STOP signal.

$$\Delta = nT_{clk} + F_{START} - F_{STOP} \quad (2.30)$$

In Figure 2.20, a waveform description of the coarse time and the fine time measurements can be seen. As it can be seen from the figure, in a typical digital TDC implementation, the coarse time corresponds to the number of clock periods counted with the system clock and the fine interval is the position of the trigger signal with respect to the clock edge within a clock period. In this figure, T_{START} represents a common start for all time events and T_{STOP} is the trigger event quantised by the TDC.

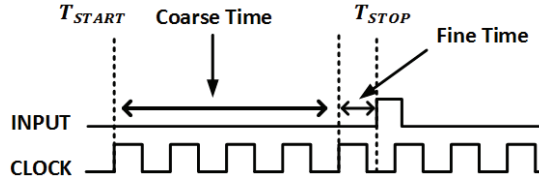


Figure 2.20: Waveforms of a Nutt Interpolation [7]

In Figure 2.21, a Nutt Interpolation architecture can be seen, illustrating how the START and STOP signals are calculated separately and combined for the final result.

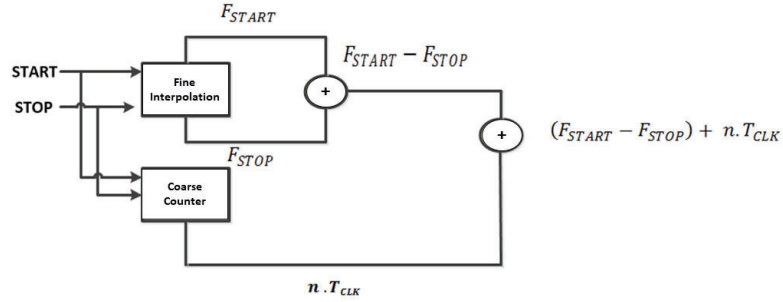


Figure 2.21: Illustration of Nutt interpolation architecture

2.5.3 Fine Time TDC Schemes

The fine time TDCs can be defined as TDC schemes whose measurement performance go beyond the limitations of the coarse time quantisation. In other words, the measurement is not limited by the system's clock period. Fine interpolators are implemented by using different technologies which include FPGAs, ASICs and analogue circuits. In this section, the implementation methods of fine TDCs and concepts related to fine TDCs will be discussed.

2.5.3.1 Time-to-Analogue Converters

Predecessors of TDCs can be ADCs. The main property that differentiates TDCs from ADCs is essentially the domains they are utilised in. However, ADCs can be used for implementing TDCs. These devices are named as TACs or Double-Conversion TDCs in the literature.

Time-to-Analogue Converters effectively use double conversion to achieve the time digitisation. The first conversion is referred to as the time's conversion to the voltage and the second conversion is the voltage to the digital by use of an ADC [86]. This is achieved by converting the time between the START and STOP signals to the electrical charge and storing this electrical charge in a capacitor. Once the STOP signal is asserted, an ADC is used to convert the time interval into a digital value [94].

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

This operation can be achieved by using an integrator op-amp. The integrator circuit is used for storing the charge over an integration time which is the time between the START and STOP events. Essentially, the integrator outputs the voltage as an integration of the input voltage over the time which is framed with the START and STOP events. Later, the output feeds the ADC, and it generates a digital value for the time interval. A circuit diagram for a TAC TDC can be seen in Figure 2.22 [94].

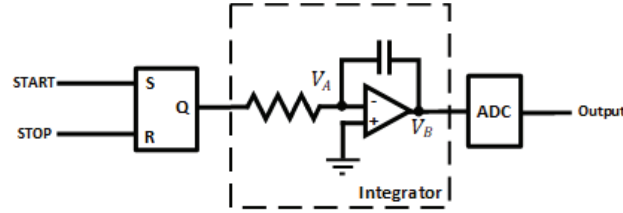


Figure 2.22: Illustration of TAC based TDC

Waveforms of how the time difference (ΔT) between the START and STOP is converted to the output code can be seen in Figure 2.23.

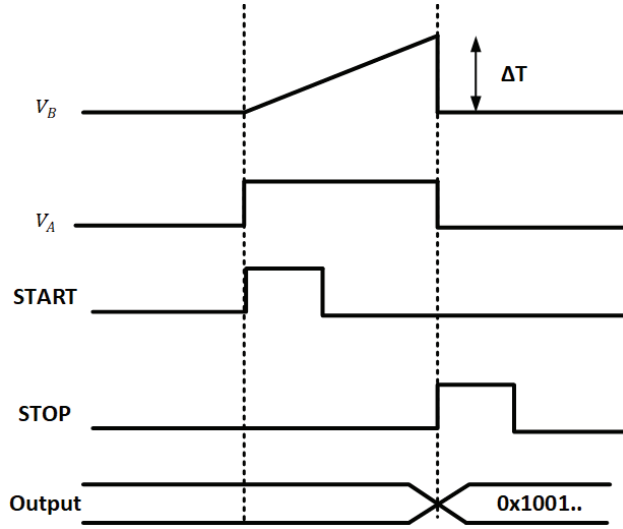


Figure 2.23: Waveforms of TAC based TDC

TAC based TDCs are generally considered as one of the oldest TDC techniques and predecessors of modern digital methods. The research conducted on TACs extends further back than the digital implementations and the resolutions achieved with these implementations vary. As it was described in [95], the resolution of 10 ps achieved by using TACs in 1988, while achieving this sort of resolution is still challenging for the most of the digital circuitry. Also there are some other TAC TDC [96, 87, 97, 98, 99] which achieved resolutions between 10 ps-312.5 ps. The main factor affecting the resolution is generally the quality of the components used in these circuits [1]. Also,

circuit layout and signal integrity also contributes resolution of the TAC TDCs, and hence, each TDC channel require careful designing and multi-channel designs are too complex and expensive to design compared to their digital counterparts [94]. One of the highest resolution single channel TAC-TDC example designed by Keranen, Maatta and Kostamovaara [100], where the resolution of 1 ps achieved.

Another limitation of these methods are the dead-time between each measurement and the limited measurement ranges. The dead-time is caused by two reasons. Firstly, the requirement of double-conversion where the charging of the capacitor and conversion of the ADC introduces separate dead-times. Secondly, the necessity of discharging the capacitor fully before a measurement. Also, since the time between the START and STOP is solely converted by using an integrator, measurable ranges are limited by the capacity of the capacitor. Typically, μ s measurement ranges are measured with these schemes.

2.5.3.2 Time Expansion Method

To increase the resolution of a TDC, time expansion methods have been proposed in a few different places in the literature. The time expansion method is an analogue method which aims to expand the interval of measurement by multiplying the signal in the time domain and dividing it into the gain of an amplifier. This is done by using a capacitor which is being charged with an assertion of the START signal by enabling the first current source. With the STOP signal, the discharging current source is being enabled, and it slowly discharges the capacitor. During this time, when the total charge of the capacitor is higher than the threshold voltage, the counter is enabled by the comparator, and it counts the time difference between the START and STOP [94].

With this method, high resolutions can be achieved by using fast counting counters. The best-achieved resolution was recorded as 630 fs [101], where the measurement range was 1.3 ns. Also, some other implementation such [102], 8.9 ps resolution was achieved for 128-channel TDC, and in [103], 3.75 ps resolution delivered by a TDC in 65 nm CMOS. However, the main limitation of this method is an introduction of large dead-times since until the capacitor is fully discharged, it is not possible to start a new measurement. Therefore, Like TAC TDCs time expansion TDCs are not ideal for high count-rate TDCs. The explanatory diagram can be seen in Figure 2.24, where the charge of the capacitor C is varied between with two current sources, initially with the START signal the current source I_1 starts charging the capacitor C and with the STOP it slowly I_2 discharges the capacitor. Since $I_1 \gg I_2$, discharging takes longer than the actual time difference. During this time, the counter is enabled by a comparator, and it keeps counting until the capacitor is fully discharged.

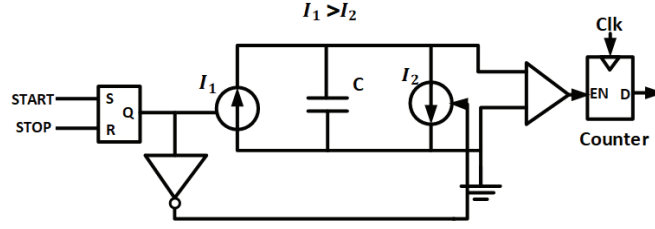


Figure 2.24: *Time Expansion TDC*

An example waveform for the time expansion TDC can be seen in Figure 2.25

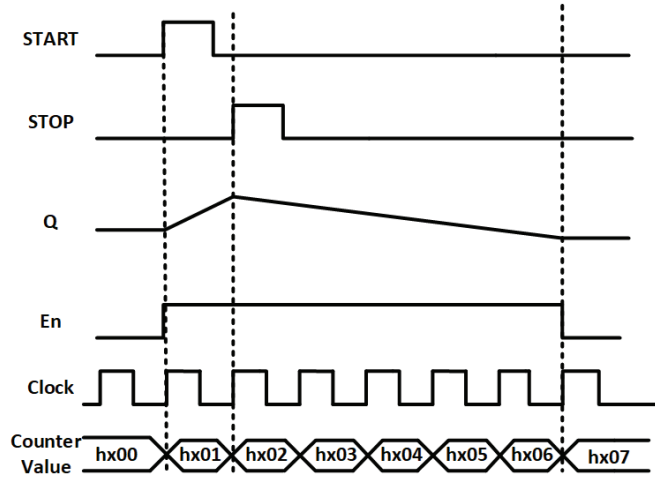


Figure 2.25: *Waveforms of the Time Expansion TDC*

The time expansion typically amplifies the time difference between the START and STOP by the gain of the amplification circuit, where T_{Start} and T_{Stop} are the time of the START and STOP events before the expansion processes, N is the quantisation multiplier required between two events, T_{res} is the actual resolution. Therefore, the time difference between the START and STOP can be as in Equation 2.31.

$$T_{Start} - T_{Stop} \leq N * T_{res} \quad (2.31)$$

Hence, for the amplified values of the START and STOP events can be rewritten as below.

$$T_{AmpStart} - T_{AmpStop} \leq N * T_{unamp} \quad (2.32)$$

By multiplying the absolute time between the START and STOP with the gain of

the amplifier (G_T) the expression can be rearranged like in Equation 2.33.

$$T_{AmpStart} - T_{AmpStop} = (T_{Start} - T_{Stop}) * G_T \quad (2.33)$$

Hence,

$$N * T_{unamp} = (N * T_{res}) * G_T \quad (2.34)$$

Finally, the actual resolution is improved by the factor of the gain as it can be seen in the equation below.

$$T_{unamp}/G_T = T_{res} \quad (2.35)$$

2.5.3.3 Tapped Delay Lines

Probably the most popular fine TDC implementation would be Tapped Delay Lines based TDCs. Taped Delay Lines consist of ideally equal length delay elements which are chained together and used for subdividing time intervals into equal bins. Typically, the START signal propagates through the delay line until the STOP signal is asserted and the state of the delay line is captured with the edge of the STOP signal. Also generally, instead of the STOP signal, the clock edge is used. Therefore, the trigger's relative position to the clock edge is used for the quantisation. When the clock edge is used for capturing the state of the delay line, effectively the clock period is divided into equal bins [1, 94].

The resolution of the TDC in delay line based methods is equal to the smallest delay element since delay elements are the smallest bins that the time intervals are split into. In practice, D type flip-flops are generally used to capture the code formed in the delay line, which is trialling '1's or '0's. This pattern is a thermometer code and requires conversion to binary code. Priority encoders are typically employed for the conversion of thermometer codes to digital codes. The conversion is done by looping through the code, and the highest '1' bit's position is converted into a digital value. Therefore, to increase the number of bits represented by the fine code, the logic consumption is required to be increased exponentially [94]. An illustration of a tapped delay line can be seen in Figure 2.26.

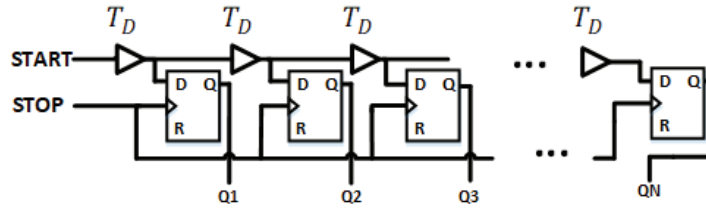


Figure 2.26: A Tapped Delay Line TDC

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

In Figure 2.27, waveforms for a Tapped Delay Line TDC can be seen, where the START signal is delayed by a signal element (T) in each delay line stage.

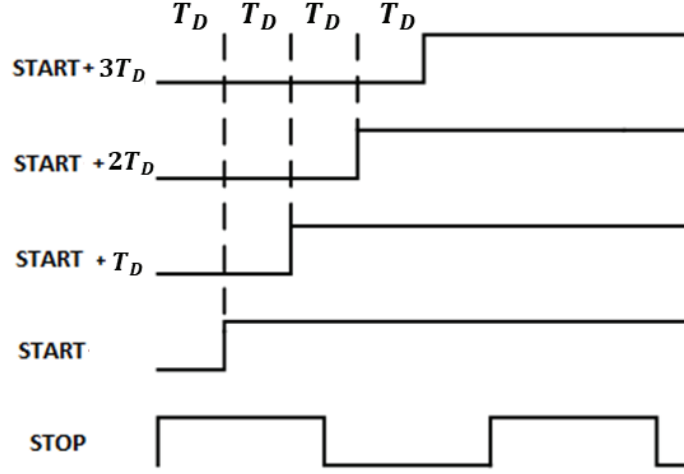


Figure 2.27: Waveforms of a Tapped Delay line TDC

A tapped delay line TDC quantises the START event with respect to the edge of the STOP event. Therefore, ideally if a delay line has N number of delay elements, each delay affecting a single bin is $T_D = T_{STOP}/N$, where T_D is a delay element. T_D is also the LSB resolution of the TDC. Timestamps for events can be formulated as below, where T_D is each delay affecting the delay element, ϵ is the quantisation error, and the K is the number of delay elements passed until the STOP signal.

$$T_{event} = T_D * K + \epsilon \quad (2.36)$$

Also, there are other tapped delay line variations in the literature, which are the Hierarchical TDC and Differential TDC. The hierarchical method mainly aims to improve logic utilisation of the TDC by using more than one delay line. Each delay line inputs another delay line in half of its length with the previous one. Therefore, a binary code is output and without using priority encoders [104]. The Differential TDC is a TDC variation which aims to improve the signal integrity of a TDC by operating in differential mode between elements by alternating the propagating signal by '1' and '0'. This method was reported to improve the metastability by in-trade of an increase in the logic utilisation about four times [104].

Tapped delay lines TDC have implemented by using both FPGA and ASIC technologies. The ASICs typically achieve better resolution than FPGAs due to customisable logic elements. However, it is possible to make high-resolution TDCs by using FPGAs. For instance in [105], with Xilinx Virtex 5 FPGA 17 ps resolutions have been achieved, while in [106] with one of the obsolete FPGAs architectures Altera ACEX1K 400 ps

resolution was reached. Also, some very high precision TDCs as it was described in [107], achieved 2.3 ps RMS resolution on Altera Cyclone V FPGA.

FPGA based delay line TDCs are commonly implemented by utilising carry chains structures located within the FPGA SLICES (Logic Elements for Intel/Altera). These SLICES are used for implementing various logical operations, hence they accommodate Look-up Tables (LUTs) for implementing logic operations between inputs, a carry chain with dedicated XOR and multiplexer logic for mathematical operations and a D type flip-flop for the memory. In TDC implementations, carry chain structures are used to cascade input triggers from Carry-In (CIN) to Carry-Out (COUT), and during this operation the internal logic of SLICE adds propagation delay at the each stage of the adder operation. The 1-bit output of the each carry element is captured by flip-flops with the clock edge and the combination of output code is used for forming the thermometer code. Moreover, carry chains can be combined between different FPGA SLICES for larger code outputs since each number of bits can be output from a SLICE is limited (4-bits for Spartan 6 Architecture). [108, 8]. An illustration of an carry chain logic within an FPGA SLICE based on Xilinx Spartan 6 architecture can be seen in Figure 4.2. In Figure : 4.2, $S_{N,0}, S_{N,1}, S_{N,2}$ and $S_{N,3}$ represent 4-bit for carry chain output for $SLICE_N$, A, B, C, D are the logical inputs for the SLICE and CLK represents the clock.

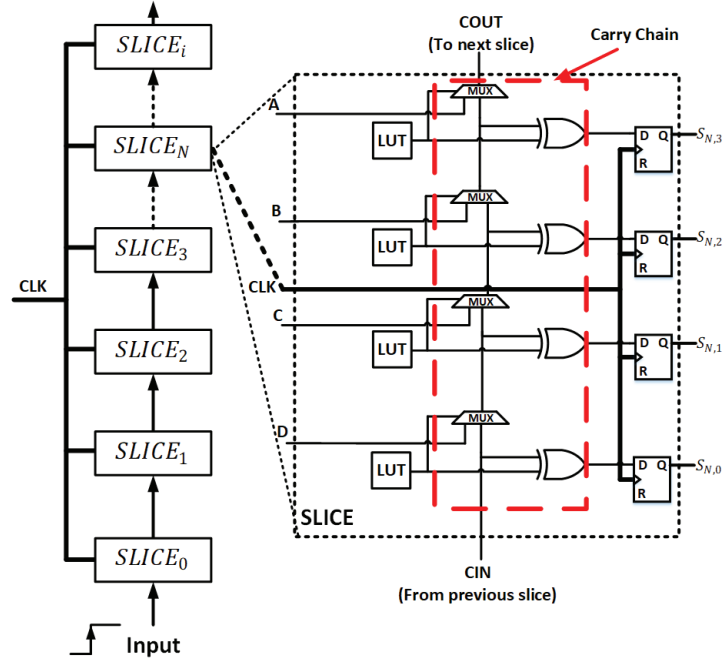


Figure 2.28: Demonstration of a carry chain within an FPGA SLICE [8]

In the ASIC side of the implementation, it is possible to have adjustable delay

elements which compensate for the changes in voltage and the temperatures across the delay line by using Voltage-Controlled Ring Oscillatorss (VCROs). The usage of VCROs is effectively done by inputting the last bin of the carry chain into a phase detector and based on the detected phase, the voltage affecting the delay element changed by a charge pump to stabilise the delay on each element [1]. This method is not possible to implement on an FPGA since VCROs are not available. So, the FPGA based implementations typically suffer from non-linear delay propagations across delay lines [109]. On the other hand, ASIC can have dedicated uniform delay lines due to its customisable nature. The high-resolution delay line TDCs can be implemented using ASIC technology. In the literature, 3 ps resolution was achieved as it was described in [110].

2.5.3.4 Vernier Delay Lines

Another delay line variation is a Vernier Delay Line (VDL) TDC. A Vernier delay line accommodates fast and slow delay lines and the difference between propagation delays affecting these delay lines are exploited to quantise the time. In the Vernier method, the thermometer code is captured when the START and STOP signals, which are propagating in different delay lines, catch each other. Therefore, the resolution is the time difference between the fast and slow delay lines' delay elements [94].

In a conventional VDL, D type flip-flops would be used to capture the state of the delay lines, and so, the thermometer code represents the time between START and STOP signals. The fast delay line's delay elements are slightly faster than the slow delay lines. The START signal propagates through the slow whilst the STOP signal utilises the fast delay line. As the moment, the STOP signal catches up with the START signal; flip-flops capture the thermometer code for the time quantisation.

Hence, the timestamp for T_{event} can be expressed as below.

$$T_{event} = (N_{STOP} - 1) * T_{STOP} - (N_{START} - 1) * T_{START} \quad (2.37)$$

In Equation 2.37, N represents the number of elements passed for each delay line, T_{event} is the timestamp for the time between START and STOP events, T_{STOP} is the delay element on the fast and T_{START} is the delay element on the slow delay line.

A diagram of VDL can be seen in Figure 2.29.

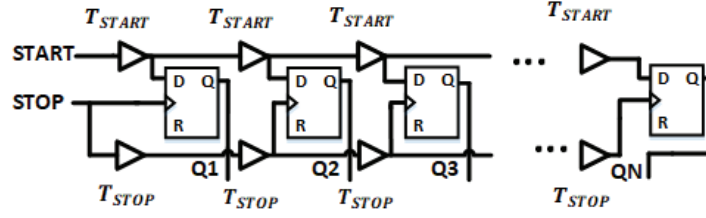


Figure 2.29: A Vernier Delay Line

Unlike a typical tapped delay line, this method utilises the difference between two different but reliable delay elements rather than the smallest possible delay elements available. Therefore, the delay difference between the fast and slow delay lines determine the resolution of the TDC, which implies with slow logic elements high resolution can be obtainable by just doubling the logic utilisation. However, in FPGAs due to the delay control and placement issues, it is challenging to achieve variable and reliable delay lines, and thus, they are challenging to implement in FPGAs. ASIC based Vernier Delay Lines can make high resolutions, as it can be seen in [111], where 880 fs resolution was achieved.

Another similar concept is the Vernier Ring TDC. Vernier Ring is a form of the Vernier Delay Line whose first and the last bins are connected to form a loop. This loop accommodates counters at each delay element to count how many times the START signal passes through the delay element until it catches up with the STOP signal. Effectively, this defines the signal position in an equivalent VDL. This method improves the logic utilisation required by a typical Vernier Line, and the resolution of the TDC would be limited to how the fast the counter can operate and the minimum difference can be maintained between the elements of two delay lines. With this method previously, 8 ps SSP was achieved with a $0.13 \mu\text{m}$ CMOS as it was described in [112]. An Illustration of a Vernier Ring can be seen in Figure 2.30, where the delay on each delay element is represented by T_d and the delay difference between fast and slow delay elements is β .

Generally, ASIC technology is preferable for implementing the Vernier method since the generation of small, reliable and consistent delay elements are possible in this platform due to the customisable transistors. FPGA platform is problematic to implement a Vernier Delay Line since it is difficult to have reliable and consistent delay elements in this platform. However, by using two different phased oscillators a Vernier method can be implemented in FPGAs, the phase difference essentially replaces the delay differences in the Vernier Delay Line.

Using the Multi-Ring Clock Oscillator Method is probably the most sensible way to implement a Vernier method in an FPGA since it is challenging to set variable and reliable delays on FPGAs. This method consists of two different clocks which are

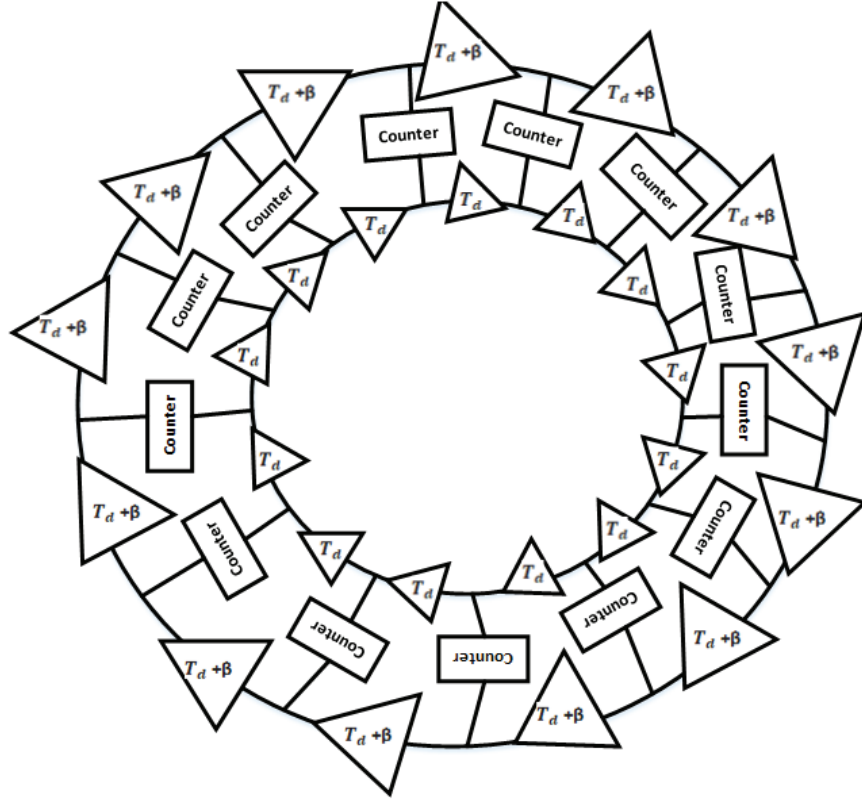


Figure 2.30: A Vernier Ring [1]

generated by two different PLLs with different phases. One of the clocks is set to a slightly faster frequency than the other, like in Vernier Delay Lines [113]. The slow clock is synchronised with the START signal, and the STOP signal is in sync with the fast clock. Since the periods of these oscillators are known, the phase difference can be used for the fine time measurement. The mathematical expression for this method can be seen below, where T_{clk} is the system clock and T_1, T_2 are clocks with different phases [9].

$$\Delta T = nT_{clk} + T_1 - T_2 \quad (2.38)$$

One clock is referred to as the fast oscillator, and the other one is named as the slow oscillator. The slow clock is adjusted to be synchronised with the rising edge of the START signal. The fast clock is set slightly faster than the slow clock and in sync with the STOP signal. The period of the fast clock can be expressed as $T_{fast} = 1/F_{fast}$ and the period of $T_{slow} = 1/F_{slow}$, where n_1 and n_2 correspond to the number of rising edges passed by the fast and the slow clocks respectively. A phase detection circuit is used to measure the phase between two clocks when the fast clock catches up with the

CHAPTER 2. RESEARCH REVIEW

slow clock, which essentially expands the time of measurement for the fine time. This operation can be expressed as the expression below [9].

$$\Delta T = nT_{clk} + n_1T_{fast} - n_2T_{slow} \quad (2.39)$$

Block diagram of a Multi-Ring Oscillator TDC can be seen in Figure 2.31.

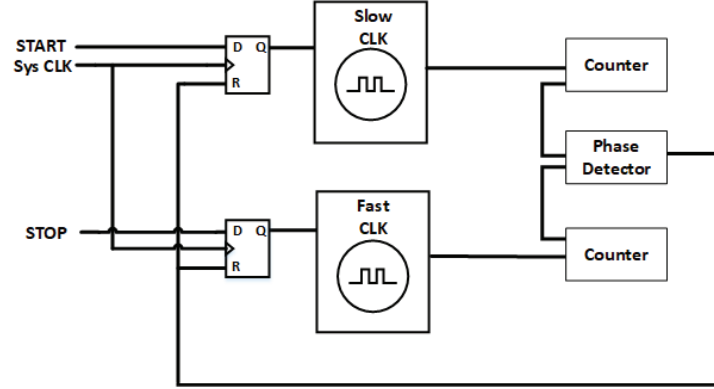


Figure 2.31: Multi-ring oscillator TDC [9]

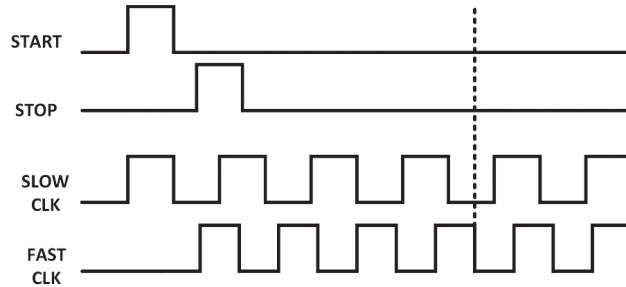


Figure 2.32: Multi-ring oscillator TDC waveforms [9]

With this method, pico-second resolutions were achieved as described in [114]. The main drawback is increased dead-time compared to tapped delay lines. The main reason for a greater dead-time is the way phase detector scheme was implemented which, is similar to time expansion TDC and expands the time for more than a single clock cycle for high resolution. However, it is still an acceptable TDC scheme used for LiDAR applications, where low dead-times are not required and as it is described in [114], 5 ps was achieved with this implementation. In addition, since no delay line is needed for this method, it improves the space efficiency for the TDC.

2.5.3.5 Pulse Shrinking TDC

Pulse Shrinking (PS) TDCs are based on continuously shrinking the START and STOP signals through pulse shrinking elements until the trigger cannot be detected any more. Hence, how many pulse shrinking elements will be used in this process is determined by the width of the pulse, and the width of the pulse is equal to the time between the START and STOP signals. Even though this type of TDC's resolution is not limited by the size of the delay line elements, due to the need of variable-sized pulse shrinking elements, the conversion time for this implementation is much larger compared to the other delay line based methods [115].

To improve the area requirement of this implementation, in a few places cyclic pulse shrinking circuits have been used [10, 116, 117]. These systems accommodate inverters which are controlled by an AND gate for activating the pulse shrinking and pulses shrink as they travel through inverters. As pulses go through the pulse shrinking component, a counter keeps track of how many times the pulse is being shrunk. However, the mismatch between processing times of pulse shrinking and counter speed is the main issue with this method. Thus, some additional delay elements are used to pipeline the counter, which results in extra cost in the area. Alternatively, in some implementations [118], Vernier Ring like pulse shrinking cycles are used, where multiple pulse shrinking elements are connected cyclically from the tip to the end. Cyclic Pulse Shrinking (CPS) TDC can be seen in Figure 2.33.

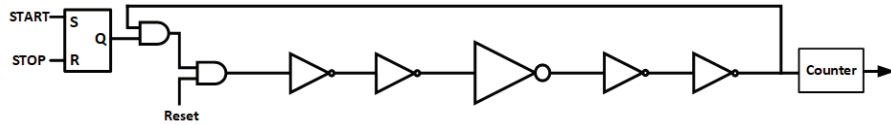


Figure 2.33: *The pulse shrinking TDC [10]*

Waveforms of the Pulse Shrinking TDC can be seen in Figure 2.34, where the time difference between the START and STOP is shrunk three times until it completely disappears. Then, the counter outputs three as a result.

Previously, in some implementations like in [115], by using row and column decoders, the required area has been tried to be reduced. Although high resolutions are achieved, low data-rates are observed due to the fact that the counter is delayed by delay lines to compensate for the timing mismatches between pulse shrinkers and counters. Therefore, the most common implementation of a Pulse Shrinking TDC is the Cyclic Pulse Shrinking TDC. In the literature, with this implementation by using customised CMOS, resolutions as high as 6 ps has been achieved [118]. Also, mostly the customised CMOS was used to implement Pulse Shrinking TDCs. The only reported FPGA based implementation can be seen in [119], where 8.5 ps resolution with 42.4 ps STD was achieved, and however, the conversion time of the implementation

was reported as 1042.1 ns. Therefore, for high count-rate implementations, where short dead-times are required, Pulse Shrinking TDC implementations are not ideal since the pulse shrinking process results in longer dead-times.

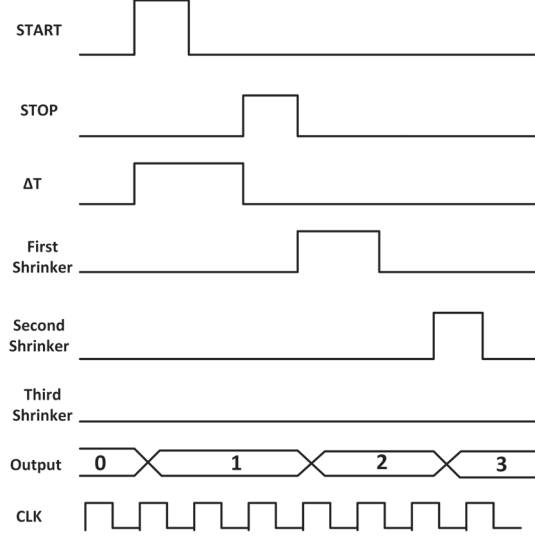


Figure 2.34: *The pulse shrinking TDC waveforms [10]*

2.5.3.6 Local Passive Interpolation

The Local Passive Interpolator (LPI) TDC method is a TDC implementation scheme which uses passive differential circuit elements to implement a TDC instead of delay elements. In this method, resistors replace logic delay elements, where they are utilised as voltage dividers between the input and output. Each series of resistors is connected in series to poles of a differential delay element as it can be seen in Figure 2.35. Since the voltage between resistors passes the threshold at different times, the output is lagged with respect to the input. By using sense amplifiers or latches the output voltage after the differential element can be captured and used for the fine time measurement [1].

Typically, 2^N resistors are needed to be represented N -bits of the code. Also, due to the utilisation of differential signals, the noise in this sort of implementation is expected to be low. However, the system suffers from the power supply, thermal and switching noises of logic elements. This results in a reduction of precision. Also, the area utilisation of this design is significantly higher than the other implementations, due to the need for a series of resistors and differential circuit elements. Also, due to its analogue nature it cannot be implemented in an FPGA.

The methods described in [120, 121, 122] can be an example of LPI TDC, where 4.7 ps resolution was achieved with LPI TDC with 90 nm CMOS. Also, by using

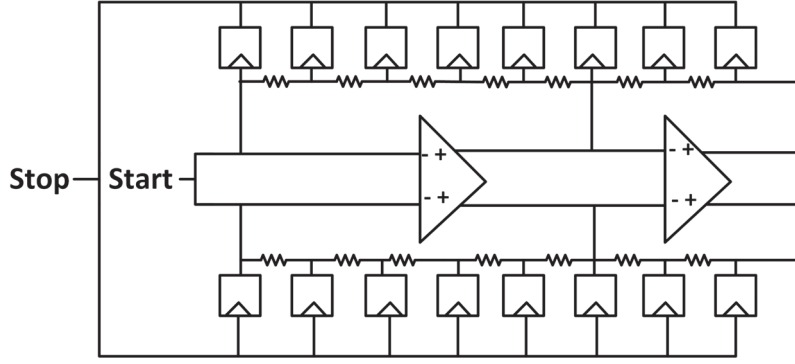


Figure 2.35: *The Local Passive Interpolator [1]*

tri-state inverters on 180 nm CMOS, 7.84 ps resolution was achieved in [123].

2.5.3.7 Stochastic TDC

The Stochastic TDC is a TDC variation which exploits voltage dither that causes metastability on flip-flops. Based on triggers position relative to the clock edge, the voltage dither of flip-flops statistically affects the output generated by flip-flops. Hence, triggers are relatively farther away from the clock edge are less likely to appear in the output code. This can be done by employing many flip-flops, and then, the time difference between the clock edge and trigger can be quantised [124, 11].

As the number of serial flip-flop increases, Gaussian distributed voltage dither errors are introduced into the time measurement increases. Each flip-flop has random voltage trigger threshold and the error on vast number of flip-flops are observed as Gaussian-distributed standard deviation σ_{vol} . The standard deviation of the measurement error σ_{error} affecting a single flip-flop can be expressed by a division of standard deviation of voltage triggering levels at flip-flops σ_{vol} to the number of flip-flops K [91, 11].

$$\sigma_{error} = \sigma_{vol}/K \quad (2.40)$$

The voltage dither affecting a flip-flop can be expressed with the Gaussian error function and can be represented as in Equation 2.41 [91]

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (2.41)$$

When the Taylor expansion is used to approximate the value x , the expression can be written as below [91].

$$erf(x) = \frac{2}{\sqrt{\pi}} \left(x - \frac{x^3}{3.1!} + \frac{x^5}{5.2!} - \frac{x^7}{7.3!} + \dots \right) \quad (2.42)$$

CHAPTER 2. RESEARCH REVIEW

To express the Cumulative Density Function (CDF) of the Gaussian Error across the flip-flops is given by as in equation below [91].

$$CDF(x) = \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x - \mu}{\alpha\sqrt{2}}\right) \right] \quad (2.43)$$

When the mean error is assumed to be zero, CDF can be expressed as below [91].

$$CDF(x) = \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x}{\alpha\sqrt{2}}\right) \right] \quad (2.44)$$

Hence, the CDF of x can be approximated to Equation 2.45 [91].

$$CDF(x) \approx \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{1}{\sqrt{2\pi}\sigma_{err}}\right) \right] x \quad (2.45)$$

For the L number of flip-flops in the chain, the CDF of the voltage dither can be arranged as below [91].

$$CDF(x) \approx \frac{L}{2} + \frac{L}{\sqrt{2\pi}\sigma_{err}} x \quad (2.46)$$

An example of a Stochastic TDC can be seen in Figure 2.36, where multiple delay lines were utilised for averaging results obtained from the voltage dithering.

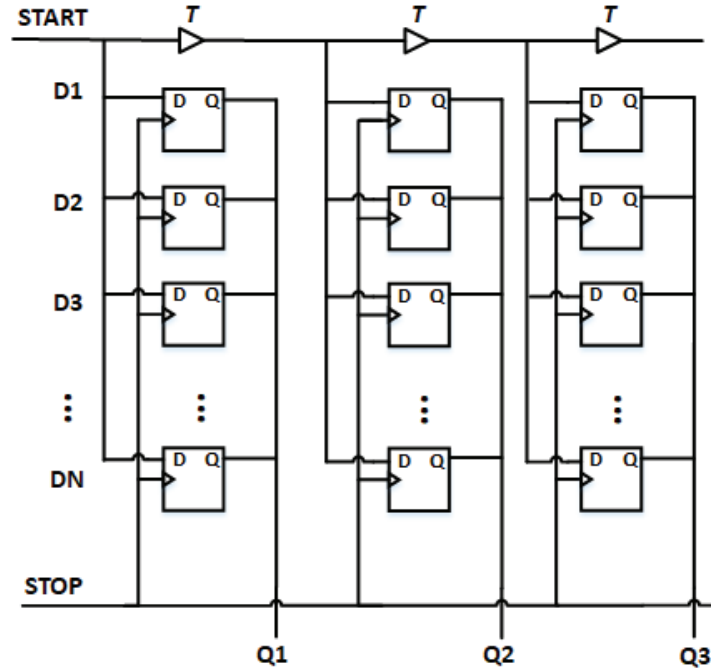


Figure 2.36: A stochastic TDC [11]

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

In this method, the resolution of the stochastic TDC can be approximated to the inversion of the CDF's slope. Therefore, the LSB equation can be seen as below, where L represents the number of parallel flip-flops.

$$LSB = \frac{\sqrt{2\pi} \sigma_{vol}}{K.L} \quad (2.47)$$

This TDC scheme has previously implemented at the University of Toronto, and 4 ps resolution was achieved [91]. Although the stochastic TDC can provide precise timings and high resolutions, it is costly in terms of the logic utilisation for long-range measurements. An increase in the number of the bits in code results in an exponential growth in the logic utilisation. Also, it is not easy to implement in an FPGA with long flip-flop chains since in case of having flip-flops in different slices will affect the precision of this scheme. Generally like in [91], ASIC is preferable for this method.

2.5.3.8 Successive Approximation TDC

Successive Approximation (SA) TDC is a scheme which makes use of the binary search algorithm for the time quantisation as it was described in [125]. The SA is done by delaying the START and STOP signals with a delay line and comparing whichever one reaches the end first. Then, the signals continue on a delay line with a shorter length. This implementation reduces the size of the required flip-flops to capture the code, and thus, the size of the logic utilisation and power consumption are reduced [1, 11].

In a typical SA TDC in each stage, the size of the delay line is half of the previous delay line. As a signal propagates, the bits of the code is decided based on whether the START or STOP signals pass the delay elements first. For instance, 8 delay elements for 4th bit, 4 delay elements for 3rd bit etc. are required. Arbitrators set the bits of the code based on testing the time difference between the START and STOP signals. If the START signal is lagging behind the STOP signal, the corresponding bit is set to a low bit, otherwise, to a high bit. In a tapped delay Line based TDC, each bit of the thermometer code requires a D-type flip-flop, and thus, 2^N number of flip-flops are needed where N is the number of bits in the code. However, SA TDC only requires N bits flip-flops to capture the code, and so, a $2^N - N$ improvement in the flip-flop utilisation is achieved [11].

As can be seen from Figure 2.37, the START signal's location in the delay line is compared with the reference signal's rising edge. With the rising edge of the reference signal, the START signal's position in the delay line is output. An example of a SA TDC can be seen in Figure 2.37. The resolution of the scheme is given by the delay affecting the single bin on the delay line which can be found by $T_\tau = T_{stop}/N$, where N is the number of bins present in a delay line and T_{stop} is the reference signal. Thus, the resolution of a SA TDC can be expressed as the minimum delay required for changing the output code [11].

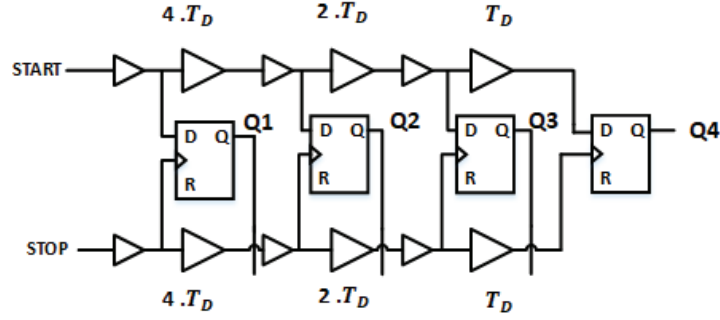


Figure 2.37: A Successive Approximation TDC

Although the flip-flop utilisation is improved, the total size of the required delay line increases as the number of bits in the code increases. For instance, a N -bit tapped delay line requires 2^N delay elements, but, for a SA TDC, the required total of delay elements is $\sum_{k=N}^0 2^k$. In order to overcome this issue, a cyclic SA TDC can be used. With this method, signals are routed in a cyclic manner, where the output of the delay line is routed back to the input. A delay element which continuously halves the delay affecting the signals is used in each round. Examples to these implementations can be found in [126, 127, 128], where the resolution range was between 1.2 ps-42 ps.

Although this method of TDC has shown an improvement in the area utilisation, the asynchronous path is difficult to design since the requirement of different sized delay elements. Thus, it is difficult to implement and achieve high precision with this kind of implementation in an FPGA due to the difficulty of customising delays in FPGAs. Therefore, this implementation is preferable for ASIC based high precision TDCs [1].

2.5.3.9 Gated Ring Oscillators

Gated Ring Oscillatorss (GROs)) TDCs are very similar to coarse counter-based TDC methods. However, this approach aims to capture the phase between measurements to apply a noise filtering effect which results in an improvement of the quantisation error. This method exploits the Nyquist criterion and achieves the first-order noise shaping [129, 9].

The N th order noise shaping circuit can be applied by combining the delay line method with ring oscillators. In order to achieve this, N number of oscillators are used, which enable counters that are connected after each oscillator. Then, the accumulated counter results are output. The enable signal to maintain the phase of the counter since it disconnects oscillators after the STOP happens. This method is also called 1-1-1 MASH $\Delta\Sigma$ TDC.

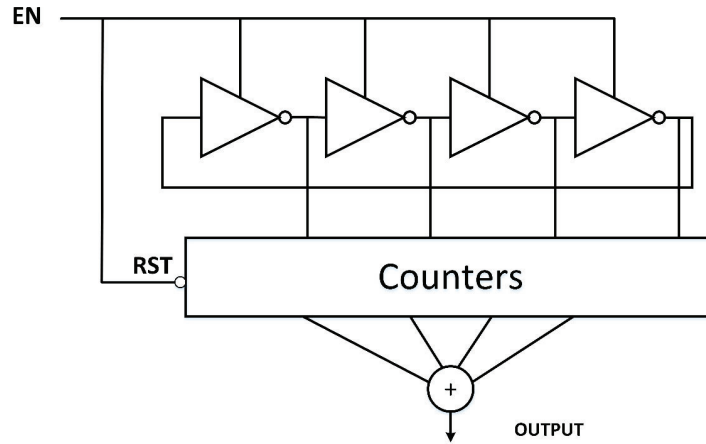


Figure 2.38: *Gated Ring Oscillators* [11] [9]

The phase is maintained by using an enable signal. Thus, when there is an input, the oscillators are enabled, and they count continuously. During this process, the phase is kept consistent from the previous measurement. The phase is kept constant by connecting and disconnecting the ring oscillators from the voltage rail by using parasitic capacitors located on the oscillators. However, as in the LPI TDC, the usage of analogue circuits results in an increase in the noise. This essentially results in an integration of the noise over the measurement time since the capacitors are not discharged during the measurement. In the literature, some examples of GRO TDCs can be [130, 131?], where noise-shaping properties of this scheme were utilised. Also, in some implementations [132, 133, 129] multi-path ring oscillators were used which improved the noise in measurements where 1 ps to 6 ps resolutions were reported.

By using a relaxation oscillator with a GRO TDC, a resolution of 6 ps was achieved in [130] by using CMOS. Also, as it was described in [134], a combination of this technique with the Vernier Method 5.8 ps resolution was delivered on a 90 nm CMOS ASIC. However, for FPGAs, this method seems not to be preferred because the FPGA fabric has predefined logic for clock oscillators and implementing a GRO TDC in an FPGA fabric is very difficult.

2.5.3.10 Algorithmic TDC

The Algorithmic TDC (Algo TDC) was initially proposed in [135], and later, it was mentioned in [136]. This method is similar to the SA TDC, but instead of reducing the delay affecting the residue, the residue is expanded for a further quantisation.

The Algorithmic TDC is implemented by using two different clock oscillators with different speeds and phases. Firstly, the time between the START and STOP is quantised by using a fast oscillator. Once the STOP signal is asserted, the slow clock is enabled to amplify the residue. Each clock trigger has a different counter, and they

are reset when the phase of the clock is wrapped around at $\phi = 0$ each time. In Figure 2.39, the measurement's waveforms can be seen, where M is the counter value for the fast clock and N is the counter value for the slow clock. At the first phase wrap of the fast clock, the quantisation is measured as $N = 1$ and $M = 4$ and for the second time $N = 1$, $M = 4$. The time framed by the red and blue lines indicates the time for the wrapping of the phase after the oscillator is switched to the slow mode [1].

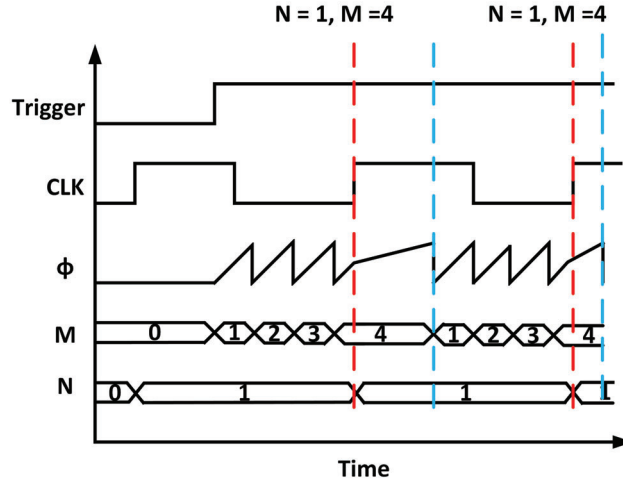


Figure 2.39: Waveforms for the Algorithmic TDC

Initially, the fast clock's counter is enabled with the START signal, and it runs until the STOP (the rising edge of the system clock). With the STOP signal, the counter stops counting and stays at the same value until its clock phase wraps around zero, while the slow clock is counting until the phase of the fast clock wraps around zero. This operation essentially amplifies the phase residue between the START and STOP signals. In each quantisation step, the residue from the previous cycle is expanded by $\frac{T_{fast}}{T_{slow}}$ [135].

By writing a recursive equation, the measurement interval can be formulated as below, where N is the value of the slow counter, M is the fast counter's value, f_{ref} is system's clock, $t_{fast_{n+1}}$ and $(\frac{f_{slow}}{f_{fast}})^n$ is the measurement error affecting n th measurement [135].

$$\Delta t = \sum_{i=1}^n \left(\frac{f_{slow}}{f_{fast}} \right)^i \left\{ \left(\frac{M_i}{f_{slow}} \right) - \left(\frac{N_i}{f_{ref}} \right) \right\} + t_{fast_{n+1}} \left(\frac{f_{slow}}{f_{fast}} \right)^n \quad (2.48)$$

In the literature, this method has been implemented on 0.35 μm CMOS, where, the error caused by the mismatch of components was around ± 2 ps. After the calibration for mismatches, the standard deviation of the TDC was measured as 0.3 ps [135]. Although, the method is a promising high-precision and low area utilisation method, the main issue with this architecture is it is a time expansion method with two

different clocks, and the dead-time is higher than the system's clock period. Thus, for the applications, where low dead-time and high count-rate are required, this is not a desirable solution [1].

2.5.3.11 SERDES TDC

Serialiser/Deserialiser (SERDES) logics are used in high-speed communication systems, where I/O channels are limited. The high-speed clock is used for serialising the parallel data of the transmitter and at the receiver's end data is de-serialised backed to the original format. This operation results in a slow data-rate input and a fast data-rate transmission. The slow data rate output is essential, preventing data loss during the transmission. An illustration of a SERDES TDC can be seen in Figure 2.40.

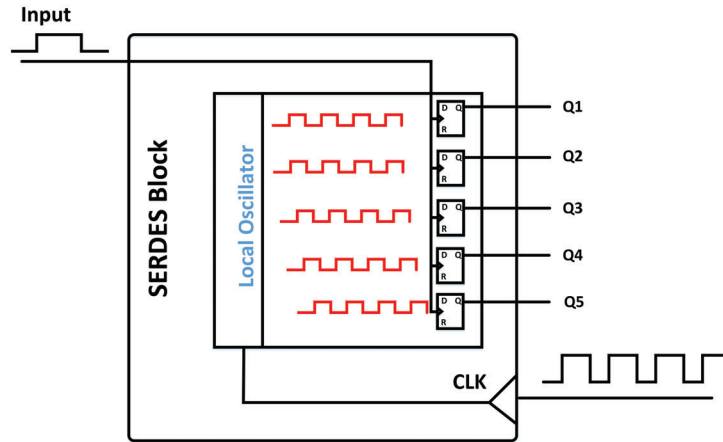


Figure 2.40: A *SERDES* based TDC

Modern FPGAs can provide SERDES blocks that can generate clocks with 10 times faster than the system clock. A SERDES block in an FPGA consists of shift registers, which are clocked by the multiplied system's clock. These shift registers can be used to quantise the system clock similar to tapped delay lines. As it was described in [137], a SERDES based 96-channel TDC was implemented on Altera Stratix EP1s30F780C6 FPGA which achieved 1.2 ns resolution. Also, as it was reported by CERN [138], newer FPGA chips (Xilinx Artix7) can achieve 10 ps resolution and 64-channels, but the precision of this implementation was reported as 150 ps. Typically, this method can be used to implement a simple TDC with a large number of channels, and however, they have shown limited precision.

2.5.3.12 Wave Union Launchers

Wave union launchers are another method of improving the precision of a TDC as it was described in [139]. A wave union launcher utilises signal trains which accommodate

multiple trigger edges generated from a single input to improve the resolution. This method aims to resolve the non-linearity issues which was referred to as "ultra-wide bins" caused by the inter-slice routing of signals in an FPGA. The method proposes to resolve this issue by sending trigger trains to be quantised multiple times in the same delay line. Two different methods have been proposed to implement this scheme.

Type A

The first method (Type A) is using a Finite Step Response (FSR) wave union launcher, where a pre-set pulse train is released upon detection of the trigger signal. Each signal inside of the pulse train is quantised separately, and results are combined later to provide precise measurements.

As it can be seen in Figure 2.41, type A consists of two sections of carry chains. The first section is referred to as K and the second section as L . Thus, after a pulse is registered in the K section then, it goes under a quantisation for the second time in the L section. In [139], K section was set to 16 bins, while L was 48 bins. Codes generated in K and L sections are combined, and this process is repeated for the entire pulse train. The resultant code is the summation of codes generated in sections K and L .

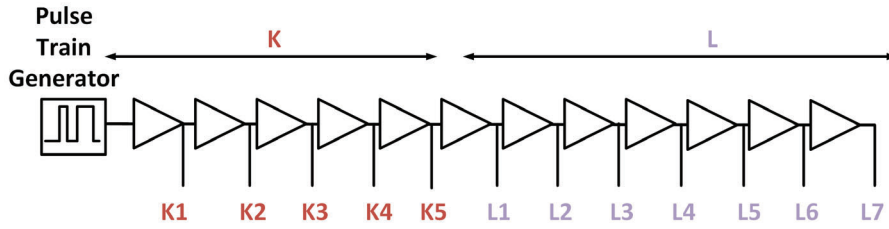


Figure 2.41: *Illustration of type a wave-union launchers*

Type B

In the second type (Type B), instead of FSR, a ring oscillator is used, and hence, this method is referred to as the Infinite Step Response (ISR) wave union launcher. The input trigger starts the oscillator, and the oscillator creates pulse trains. It should be noted that this oscillator stays active for multiple clock periods. Also, the ring oscillator should be asynchronous to the system clock to avoid hitting the same bins all the time.

The type B wave union launcher forms three different types of jump patterns from ring oscillator periods. According to [139], there are 0, 1, 2 number of oscillator period jumps which form jump patterns U, V and W respectively. Code values determine these jump patterns for S bins, and these jumps can be listed as below :

- If the output is in the range of $3S/4$ to S and proceeded by $S/4$ to $S/2$, it means both signals were corresponding to the same oscillator's edge. This sort of pattern is referred to as the U pattern.

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

- If the output's value is in the range from $S/4$ to $S/2$ and proceeded by $3S/4$ to S , it means the oscillator is faster than the system clock. Hence, two oscillator edges have been missed. This sort of a jump pattern is referred to as the W pattern.
- The other jump patterns which missed one ring oscillator period are referred to as V patterns.

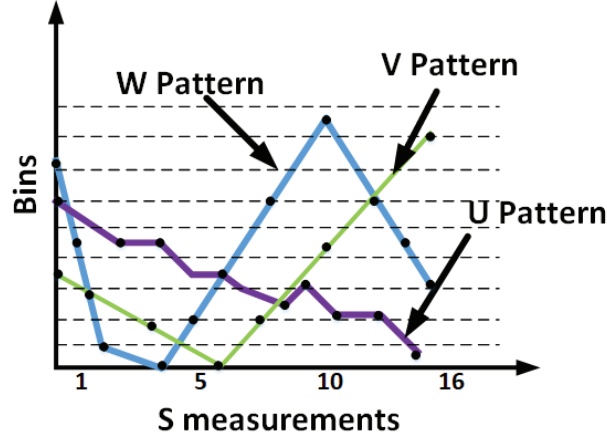


Figure 2.42: Illustration of jump patterns

The jump patterns are related to the phase of the oscillator, which generates the input trains. The ring oscillator's frequency is set to slightly faster frequency (T_{osc}) than the system clock (T_{clk}). The time for the input at n th clock (T_n) can be formulated as below, where t_0 arrival of time.

$$T_n = t_0 + n.T_{osc} \quad (2.49)$$

To characterise the jump patterns related to the phase, 16 measurements are taken with the system clock in [139]. These 16 measurements are represented with the letter s and the arrival time for s time is T_S , and where, s is between 0 to 15.

$$T_n = T_S(s) + s.T_{clk} \quad (2.50)$$

The final equation can be arranged as below.

$$t_0(s) = T_S(s) + s.T_{clk} - n.T_{osc}(s) \quad (2.51)$$

CHAPTER 2. RESEARCH REVIEW

The equation given above is also modified in actual implementation based on jump patterns, where Du , Dv and Dw represent the average calculation applied for each pattern.

$$t_0(s) = T_S(s) + T_L(s) \quad (2.52)$$

$$T_L(s) = T_S(s-1) + \begin{cases} T_{clk} = Du & \phi = U \\ T_{clk} - T_{osc} = Dv & \phi = V \\ T_{clk} - 2.T_{osc} = Dw & \phi = W \end{cases} \quad (2.53)$$

The results for 16 measurements go under a low pass Infinite impulse response (IIR) to produce an averaged calculation for each trigger. This averaging results in a delay correction and the delay correction process $t_0(s)$ had an almost constant value for every measurement for the same trigger. For further improvement in the resolution, these 16 measurements are averaged again.

With the first method, a significant improvement in precision has been achieved. The original 165 ps worst-case and 60 ps average resolution has been improved 65 ps worst case to 30 ps average resolution. While improving the resolution, the dead-time was only increased from 2.5 ns to 5 ns. The second method mainly improves the RMS error affecting the measurement, and the improvement was observed as 40 ps to 10 ps in 16 measurements. However, the second time had a much greater dead-time increase which was 18 times (2.5ns to 45ns) [139].

Wave union methods showed the capability of being a valid FPGA based TDC method. However, the complexity of the implementation is more than other precise TDC methods such as TDLs or VDLs. Also, both Type A and Type B seemed to provide dead-times longer than the clock period due to the usage of pulse trains, which hinders the optimal count-rate of the operation. Hence, it is not ideal for high count-rate required systems.

2.5.3.13 Digital Signal Processor Based TDC

As it is described in [140], one method of implementing a TDC is using a DSP. A typical FPGA based TDC uses carry chains or LUTs for building delay lines. However, these methods are proven to have non-linearity issues which result in non-optimal propagation delays. Alternatively, on-chip DSP block (DSP48A1) is proposed to be used to implement an optimal delay line by using the carry logic located in a 48-bit dedicated adder. Each DSP48A1 block in Xilinx Artix-7 FPGA was reported to be providing 800 ps for a 48-bit code which corresponds to around 17 ps average delay, while each CARRY4 element in the same architecture provides 20 ps average delay. Although the total amount of delay on a DSP block was shorter than carry chains equivalent available on an FPGA, delays seemed to be accumulated in specific bins, and thus, could not be used as a TDC due to severe non-linearity [1].

However, in a follow-up paper described in [141, 1], by using a population counter and input offsets to alter the behaviour of the DSP block, this implementation was possible. To reorganise the bins in increasing order, population counters are used. Effectively, a population counter reorders bins to fix the linearity problems caused by non-equal bin widths. By adding offsets to 4 delay line blocks, it was managed to bypass the third of the large bins. By summing the final results generated by each block, the final output is formed. With this method, 5.25 ps resolution has been achieved on Artix-7 FPGA [1].

The main disadvantage of this method is the need for a population counter to reorganise the bins, which introduces operational complexity and further dead-time to the encoding process of the thermometer code. Therefore, it could result in an additional overhead on top of the TDC calibration and the linearisation in an integrated time correlator system, where the logic utilisation will be typically high [1].

2.5.4 Measurement Errors and Precision of a TDC

A quantisation of time and its digitisation inherit inevitable measurement errors which affect the precision of the timing measurement. The concept of measurement errors in ADCs reflects the measured sinusoidal input and spectrum of the output signals, which is obtained by applying the Fourier Transform to the input to transfer it to the frequency domain. It takes multiple measurements to form the spectrum, and a process of constructing this spectrum is called the single-tone experiment. This spectrum is used to obtain Signal-to-Noise Ratio (SNR), which reflects the noise effect in the output signal and Signal-to-Noise and Distortion Ratio (SNDR), which expresses the noise, non-linear distortion and harmonics affecting the output signal [94]. Similar to the single-tone experiment in ADCs, the single-shot experiment is adapted to replace the precision of the TDC since it represents a time interval rather than a sinusoidal signal. The single-shot experiment consists of measuring a fixed time interval in multiple instances and characterising the measurement ranges and errors affecting these measurements. These errors can be listed as the offset & gain errors, TDC's non-linearity, quantisation errors and metastability problems. Under this section, the errors affecting TDCs and the precision of TDCs will be discussed [94].

2.5.4.1 Precision Parameters of a TDC

The precision of a TDC determines the error margin of each measurement taken by the TDC. Typically, the measured fixed interval is referred to as ΔT , which is the difference between the START and STOP signals of the TDC. Conventionally, like as it was explained as a LiDAR technique in section 2.2.1.3, the histogram is used for measuring distances through a correlation of multiple instances and around one large histogram bin, a Gaussian peak is observed as a result of jitter and noises affecting

CHAPTER 2. RESEARCH REVIEW

the measurements. Parameters of the precision are used to characterise the Gaussian peak formed by the distribution of measurements in a histogram.

Four different parameters of the precision can be listed as the standard deviation, single-shot precision and Full-Width at Half-Maximum.

The standard deviation: STD is a measure of the spread of values around the mean in taken measurements. They are generally shown with σ symbol. In the context of TDCs, the measurements between STARTs and STOPs form the STD expresses a Gaussian distribution and the spread on either side of the Gaussian peak. In a preciser TDC a smaller STD is expected. The STD can be formulated as in equation below [142].

$$\sigma = \sqrt{\frac{\sum(x - \bar{x})^2}{N}} \quad (2.54)$$

In Equation 2.54, where x is the samples, \bar{x} is the mean and N number of samples.

The single-shot precision (SSP): The SSP refers to the precision of the measurements obtained from the single-shot experiment. SSP essentially expresses the STD of measured values. As it was described in [143], it is measured by the division of standard deviation to the square root of two (RMS), and so, $\sigma/\sqrt{2}$. It is also known as the RMS time-resolution [143]. On the other hand, Henzler in his book [94], refers to SSP as the STD of a single-shot measurement. To avoid confusion, in this thesis, SSP is used as an RMS of STD in the single-shot measurement.

Full-Width-Half-Maximum: The FWHM is a measure of describing the extent between two extreme values in the Gaussian Distribution, which implies the width of the Gaussian peak. This width is defined by the max and minimum values where they intersect the half of the peak's ($h_{max}/2$) amplitude. FWHM can be formulated for a Gaussian plot like in the equation below [144].

$$FWHM = 2\sqrt{2\ln 2}\sigma \quad (2.55)$$

This equation also implies FWHM is approximately 2.355σ . This data provides a spectrum of data from measurements. The preciser TDCs are expected to give narrower FWHMs.

2.5.4.2 Offset and Gain Errors

Offset error is referred to as an error that results in measured codes to be shifted along the time axis [94]. Ideally, the first step of a TDC's transfer function should be equal to the LSB code of the TDC ; T_0 (the initial code) = T_{LSB} (LSB code). However, the initial

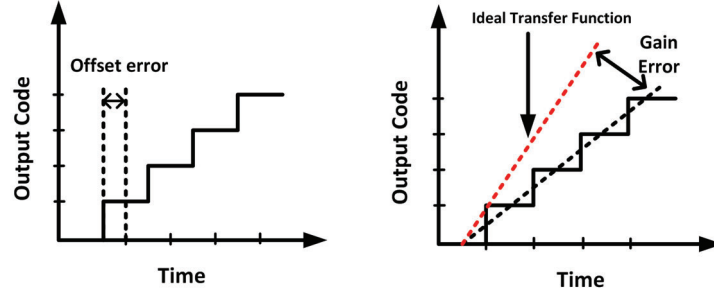


Figure 2.43: Illustration offset and gain errors of a TDC

step could be observed as deviated from the ideal initial code for multiple code steps. The equation for the offset error (ϵ) can be defined as below [94].

$$\epsilon_{off} = \frac{T_0 - T_{LSB}}{T_{LSB}} \quad (2.56)$$

Offset errors are typically caused by non-linearity issues affecting the transfer function of the TDC, which will cover in later sections. However, gain errors affect the slope of the transfer function and its ratio between the output and input code in units of LSBs. Ideally, this ratio is expected to be $1/T_{LSB}$. After the offset error is removed, the gain error can be formulated as below [94].

$$\epsilon_{gain} = \frac{T_{last} - T_{first}}{T_{LSB}} - (2^N - 2) \quad (2.57)$$

Where the T_{last} is the final code in the transfer function and T_{first} is the initial and N is the number of bits. Both gain and offset errors are affected by the linearity of the TDC and the linearity of the TDC will be discussed in the following section. An illustration of offset and gain errors can be seen in Figure 2.43.

2.5.4.3 Linearity of a TDC

In an ideal fine Time-to-Digital converter, the relationship between the input and output code should be linear. Therefore, each input should have a unique value and equal steps between adjacent codes. However, due to the various reasons such as routing, temperature and power fluctuations, TDC suffers non-linearities between the input and output. Hence, when the clock period is sub-divided into bins, the bin widths are not uniform, and this leads to missing codes in the transfer function of the TDC. TDC calibration methods are used to provide a bin-by-bin calibration for the transfer function to reduce the effects of the non-linearity.

When the converter is fed by a signal orthogonal to the system clock, by counting the occurrence of codes, the relationship between the bins with each other can be observed. An illustration of ideal and realistic bin counts can be seen in Figure 2.44.

CHAPTER 2. RESEARCH REVIEW

Since, if all the bins had equal width, every bin would have the same possibility of getting a hit. However, due to non-linearities, bin counts are observed as in Figure 2.45.

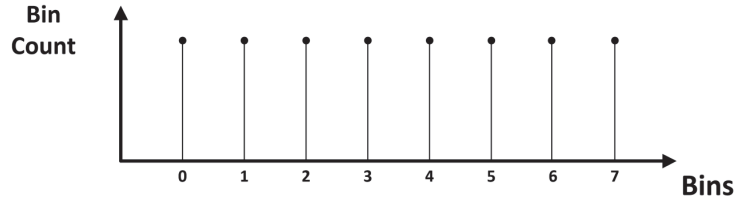


Figure 2.44: *Illustration of ideal delay line bins [12]*

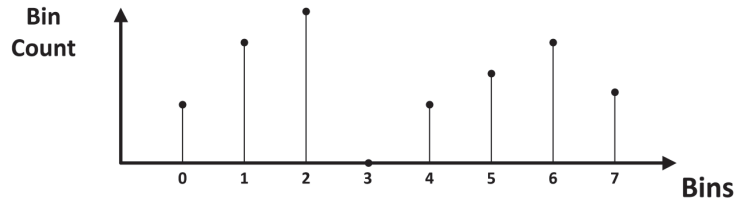


Figure 2.45: *Illustration of non-ideal delay line bin [12]*

An example of an ideal TDC transfer function can be found in Figure 2.46.

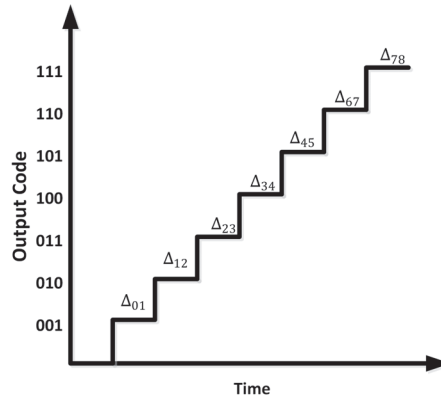


Figure 2.46: *Illustration of an ideal TDC transfer function [12]*

The non-linearity of a TDC is typically measured by using INL and DNL. In the following section, these two concepts will be discussed.

2.5.4.4 Integral Non-Linearity (INL) and Differential Non-Linearity (DNL) of a TDC

INL and DNL are the phenomenons used for describing the linearity of the TDC. An ideal transfer function of a TDC would be a straight line. The DNL represents the deviation between the ideal code and the actual code for each bin. The INL is a term used for describing the deviation between the real transfer function and the ideal transfer function. In other words, INL is the deviation between the actual transfer function and the straight line [145].

To reduce the measurement errors affecting the converter, the DNL of each code should be kept at the minimum. Therefore, some TDC schemes aim to improve this aspect of the designs. The unit of the DNL and INL is LSB, which reveals information about the deviation of codes for each step. Thus, $DNL > 1$ for a bin implies a missing code between two adjacent bins.

Since DNL is how far each bin is off from the ideal bin width. DNL for each bin can be calculated like in equation below, where T_{bin} is the actual and T_{LSB} is the ideal bin widths.

$$DNL(i) = \frac{T_{bin}(i)}{T_{LSB}} - 1 \quad (2.58)$$

Once the DNL is calculated, the INL can be measured by applying the CDF to the DNL and can be formulated as below.

$$INL(i) = \sum_{n=0}^i DNL(n) \quad (2.59)$$

The demonstration of INL and DNL can be seen in Figure 2.47.

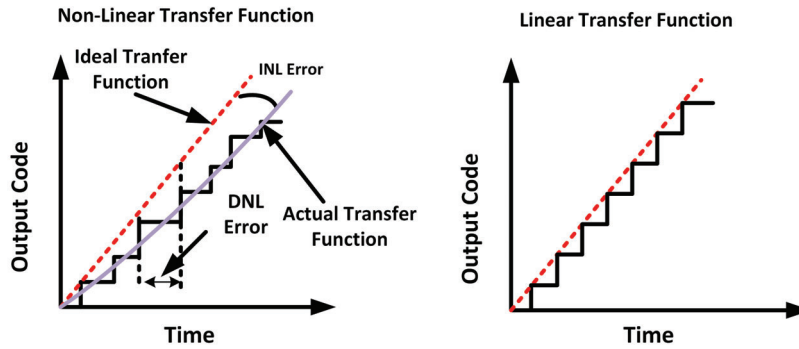


Figure 2.47: The Non-linear vs Linear transfer function of a TDC

Other phenomena significantly affect the linearity of the TDC is the number of missing bins (zero-hit bins). These bins are the delay line bins which cannot be used for the time quantisation due to the mentioned non-linearity issues. Thus, they lead to

missing codes in the transfer function of a TDC, which are observed as DNL and INL errors. Therefore, TDC can be linearised by increasing the number of non-zero codes in the transfer function. These methods will be discussed in the further chapters of this thesis. An illustration of missing bins can be seen in Figure 2.48

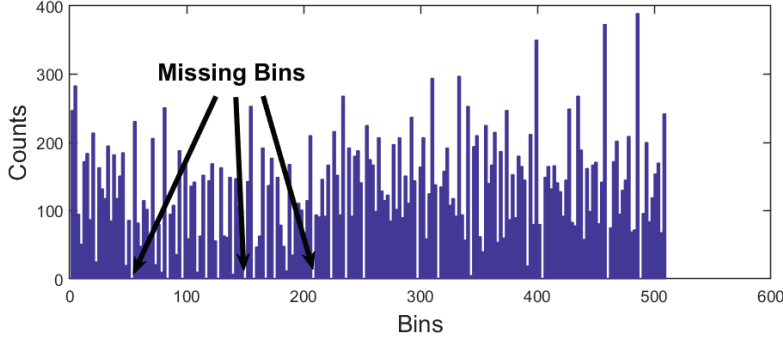


Figure 2.48: *Illustration of missing bins in a histogram*

2.5.4.5 Quantisation Errors

The quantisation errors affecting each measurement of a TDC are important phenomena needed to be discussed to understand the quantisation. In TDCs, when the measured time and actual occurrence of the time are different, this phenomena called the quantisation error. Quantisation defines values in a discrete domain, and thus, some measurement errors are inevitable. If the actual time of event is represented as T_e then, the measured time of an event can be represented as $[T_e]$. Hence, the quantisation error can be expressed as below [146, 11].

$$\epsilon = T_e - [T_e] \quad (2.60)$$

Ideally, the quantisation values change half a value up and down with respect to T_e . Hence, each code can have $\pm T_{code}/2$ quantisation error. Since quantisation values are limited in each measurement step, and each value has an equal probability of an occurrence, the likelihood of an error can be expressed with the uniform distribution [146, 11].

The STD estimation of the uniform distribution can be given as below, where the $\pm l$ is the uncertainty.

$$\sigma = \frac{1}{\sqrt{3}} \times l \quad (2.61)$$

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

For each quantisation step T_{code} . Thus, the range would be $T_{code} \pm /2$ and the uncertainty of a single measurement can be estimated as below with the quantisation step T_{code} when there is no quantisation noise is affecting the converter [146].

$$\sigma_{err} = \frac{T_{code}/2}{\sqrt{3}} = \frac{T_{code}}{2\sqrt{3}} = 0.289T_{code} \quad (2.62)$$

However, since the time interval T_e can be defined with the START and STOP signals, which are asynchronous to the system clock, the quantisation error (σ_{err}) affect both START and STOP signals. As a result, a higher total uncertainty (σ_t) can be expressed as below [146].

$$\sigma_t = \sqrt{\frac{T_{code}^2}{12} + \frac{T_{code}^2}{12}} = \frac{T_{code}}{\sqrt{6}} = 0.408T_{code} \quad (2.63)$$

The quantisation error is typically affected by random quantisation noise, and this results in an increase in uncertainty (σ_t). Thus, the quantisation error expands and can spread the error margin to other quantisation intervals. Therefore, σ_t is affected by the quantisation noise, and as the uncertainty increase, the quantisation errors can be modelled with the Gaussian Distribution [146]. When the quantisation error is determined as $T'_e = T_e - kT_{code}$, the probability distribution of T'_e being between quantisation interval $kT_{code}/2$ and $kT_{code}/2 + 1$ can expressed as below, where the original derivation was provided in [94].

$$P(T'_e) = \frac{1}{2} \left(\operatorname{erf} \left(\frac{(n + \frac{1}{2})T_{code} - T'_e}{\sqrt{2}\sigma_t} \right) - \operatorname{erf} \left(\frac{(n - \frac{1}{2})T_{code} - T'_e}{\sqrt{2}\sigma_t} \right) \right) \quad (2.64)$$

From the Equation 2.64, the Probability Density Function (PDF) of quantisation errors and how it affects quantisation intervals can be plotted as below.

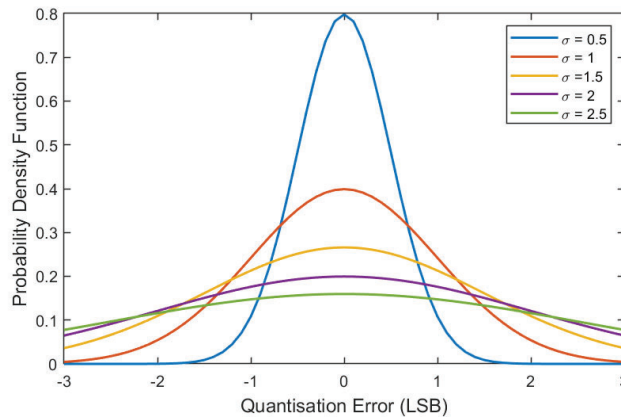


Figure 2.49: The TDC Quantisation Error vs PDF

Typically, the quantisation errors can be minimised by averaging methods as it was formulated in [146] and can be seen at Equation 2.65. In Equation 2.65, the uncertainty after averaging (σ_t') is improved by the square root of the number of measurements taken (N). The derivation of this equation can be found in [146].

$$\sigma_t' = \frac{\sigma_t}{\sqrt{N}} \quad (2.65)$$

In [94], the relationship between SSP and the quantisation error shown to be linear. Therefore, the increase in the quantisation error results in a degradation in the SSP. Hence, the quantisation noise is needed to be kept minimum for a high precision. The noise sources can be physical, substrate, power supply or any cross-talk noises affecting the system, temperature and voltage fluctuations. Therefore, it should be understood that the precision of a TDC is not limited by the size of the smallest delay element but, also the noise affecting the system.

In addition, as it was described in [94], the process variations affect the quantisation noise of the system and change the precision. These process variations can be global variations such as power and temperature noises affect but also local process variations such as comparator noises, clock skews and buffers trees. These variations affect the linearity, quantisation and offset errors of the TDC and calibration is intended to overcome these problems. Moreover, local process variations individually vary between devices used in the system and the larger systems suffer from more substantial local process variations.

In the literature, as it was described in [147], an averaging method using multi-chain measurement and averaging using both clock edges has been used to improve the quantisation noise. Also, as it was mentioned in section 2.5.3.9, MASH GRO TDC [129] was shown to improve the quantisation noise.

2.5.4.6 Metastability of a TDC

Metastability is a phenomenon that occurs due to setup and hold time violations of flip-flops and results in a flip-flop proceeding into a state which produces unpredictable outputs [148]. Due to the asynchronous nature of TDCs, they inherit metastability problems. Typically, a TDC accommodates a set of flip-flops to capture the quantisation of events. When each flip-flop has T as the time to process the trigger, readout the event and store it into the memory and t is the event of the trigger, if the $T + t$ is happened to be in the critical window of a flip-flop, this will result in an unpredictable output [149]. An example of flip-flop metastability can be seen in Figure 2.50, where T_{CW} is the critical window, T_{SU} is the setup time, and T_H is the hold time of the flip-flop.

The metastability is reduced when the number of delay stages and the synchronising flip-flops increases, and hence, the resolution increases. However, the increase the

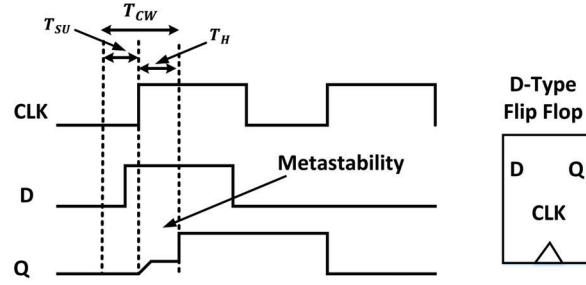


Figure 2.50: *Illustration of flip-flop metastability*

delay stages in a TDC results in an increase in a latency. Therefore, there is always a trade-off between the metastability and latency [150]. The Mean Time Between Failures (MTBF) measures the probability of metastability in flip-flops, and it can be formulated as below.

$$MTBF = \frac{e^{M/\tau}}{T_w \times F_{clk} \times F_i} \quad (2.66)$$

In Equation 2.66, where M is the time spared for the metastability, τ is the time resolution of the flip-flops, T_w is the critical window, F_{clk} and F_i are the clock and input frequency [150].

Typically, the metastability issue can be addressed in three different ways. The first way is increasing the dead-time of TDC and giving more time to metastable bits to stabilise [149]. Adding more dead-time will essentially increase the readout time of the TDC [149]. Secondly, a more synchroniser logic could be employed for the TDC. This can be done by adding more flip-flops and delay elements into the TDC. This essentially increases the M in the Equation 2.66 [150]. Alternatively, the speculative computation approach can be used to tackle the metastability as it was described in [151, 152]. For instance, in a system where synchronisers are guaranteed to have metastability in the output, the metastability can be resolved after the computation by adding a resolving time. This method also allows the usage of an adaptive resolving time by knowing the metastabilities of the previous computations. Also, Grey-code counters can effectively reduce the metastabilities affecting coarse counters as it was described in [153] by changing only a bit of the counter at a time.

2.5.5 TDC Calibration & Linearisation Techniques

In section 2.5.4, errors affecting the operation of a TDC, and how the precision is affected by errors have been discussed. In the literature, many architectures have been developed to reduce the effects of these errors and improve precision. These methods can be separated into two categories; linearisation and calibration techniques.

The linearisation of a TDC can be defined as techniques used to improve the DNL of the TDC by modifying its measurement method. Typically, these methods aim to linearise the probability of recording a hit of each code's bin in a histogram. On the other hand, calibration methods do not change the measurement method, instead as a post-processing step, calculates corrected codes for a bin-by-bin calibration. Therefore, calibration aims to improve the INL of the TDC since it forms a calibrated transfer function based on bin widths.

Under this section, the TDC calibration and linearisation techniques will be discussed. Some of these techniques are also reviewed and published in [1].

2.5.5.1 Direct Calibration

Probably the simplest way to calibrate a delay line is using a direct calibration. This method uses adjustable delays to manually characterising the width of each bin of the delay line by changing the delay effecting the input. This process starts by setting the delay affecting the input value, which will result in a generation of the least significant bit in the code. The delay is increased slowly while the change in the code's digits is observed. Recording the delay required to change a bit in each code exhibits the width of each bin. However, it should be noted that it is necessary to have a delay generator which is capable of generating input delays smaller than the delay value represented by the LSB of the code. This method is a fairly simple approach to calibrate a TDC delay line since it is only done by recording the delay value whenever code changes one value to another. However, this method would be very exhaustive in high-resolution TDCs due to the usage of long delay lines [154, 1, 155].

2.5.5.2 Code Density Testing Calibration

One of the standard delay line calibration methods in TDCs is the code density testing calibration. This method is based on testing the bins of the delay line with an input signal, which is orthogonal to the system clock and counting the number of hits each bin is getting. By dividing the number of hits at the specific bin to the total number of hits that delay line gets, gives the relative width of the bin to the other bins. This happens since larger bins are statistically more likely to get a hit [156, 14].

To calibrate a TDC, the statistical distribution of TDC bins across the delay line can be used. For this calibration method, bin counts are recorded for testing the delay line with a signal asynchronous to the system clock. By applying the CDF, a transfer function for the delay line can be formed. The CDF is done by cumulatively summing the distribution of the counts and recording the counts for each bin.

An expression for the bin width can be derived as below, where $N_{cumulative}$ is the

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

cumulative counts for each bin and N_{Total} the total number of counts.

$$Bin_{Width} = T_{clk} \times \frac{N_{cumulative}}{N_{Total}} \quad (2.67)$$

By applying the CDF a transfer function for the TDC can be formed. The transfer function of a TDC can be formulated as below, where H is the transfer function, N is the number of delay line elements for the i th bin.

$$H(i) = \sum_{i=0}^N Bin_{Width}(i) \quad (2.68)$$

In Figure 2.51 the calibration's affect on a non-calibrated transfer function can be seen.

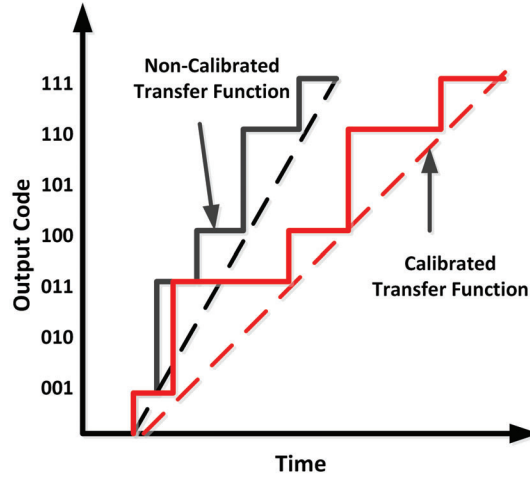


Figure 2.51: *Calibrated versus non-calibrated TDC transfer functions*

To provide a high confidence level analyses of the bin width distribution, the number of random triggers recorded should be sufficient. As it was described in [157, 155, 1], the number of hits required to provide high confidence level can be formulated like in equation 2.69, where A is the number of bits in time code, z , a is the area under the Gaussian distribution and β is the tolerance level [155].

$$M = \left(\frac{z_a/2}{\beta} \right) x (2^A - 1) \quad (2.69)$$

For example, to achieve confidence level of 97% ($\alpha = (1 - 0.97) = 0.03$), where the tolerance level of 10 % ($\beta = 0.10$) and for the 10 bit code, 481,760 histogram hits are needed [155].

The theory behind this method will be discussed in more depths in Chapter 4 since it was the calibration method used in the proposed system.

2.5.5.3 Equivalent Coding Lines

Another method to mitigate the effects of non-linearity is employing the Equivalent Coding Lines (ECLs) method. As it was proposed in [158], the ECL method uses multiple delay lines, which are characterised and calibrated using methods like the code density testing calibration, and the combination of N number transfer functions to form one equivalent transfer function. Delay lines used in this method are referred to as the Timing Coding Lines (TCL). For instance, when two delay lines are used TCL_1 and TCL_2 , their transfer functions are $H(i)_n$ and $H(i)_m$ consecutively, and then, the ECL transfer function can be represented as $H(i)_{m+n-1}$.

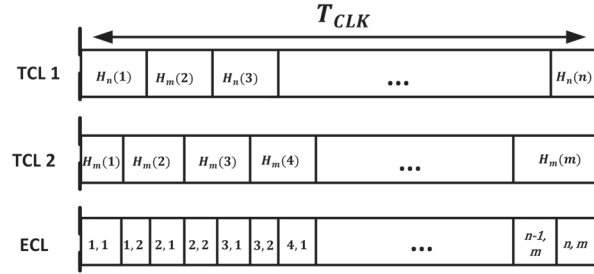


Figure 2.52: Illustration of the Equivalent Coding Lines

The idea behind ECL is when two separate delay lines are used, they are essentially unique, and the same code will have unique code representations in different delay lines due to their different propagation delay spreads. Both delay lines use the same system clock (T_{clk}) to register the state of the TCLs. The resolution of the final code achieves approximately twice as the previous one. TCL_1 achieves the resolution of T_{CLK}/n and TCL_2 , T_{CLK}/m which is approximately same as the TCL_1 . On the other hand, the ECL's resolution achieves $T_{clk}/(m+n-1)$ which is approximately equal to $T_{clk}/2$.

The quantisation of time represented as the TCL_1 and TCL_2 can be formulated as below.

$$H(i)_n = \sum_{i=1}^m H(i) \quad 0 < i \leq n \quad (2.70)$$

$$H(j)_m = \sum_{j=1}^n H(j) \quad 0 < j \leq m \quad (2.71)$$

The individual quantisation steps equivalent code from a combination of these two transfer functions of $m+n-1$ steps can be obtained as below.

$$H(i,j)_{m+n-1} = y.H(i)_n + (1-y)H(j)_m \quad (2.72)$$

where,

$$y = \sum_{i=1}^n H(i)_n \leq \sum_{j=1}^m H(j)_m \quad (2.73)$$

y is predicted to be 1 when the captured code by the first delay line is smaller or equal to the obtained code by the second delay line, otherwise, y is taken as 0.

This method was implemented on Spartan6 LX75 chip using 16 TCLs. The bin width was set to 1.14 ps (1 LSB), and after the corrections, the INL was achieved as ± 0.12 LSB (0.14 ps), and the mean resolution of 2.87 ps with the best SSP of 5 ps was achieved [158]. The equivalent resolution achieved with the method can be expressed as below from the variance of the bin widths (B_{eq}) for the clock period (T_{clk}) [158].

$$B_{eq} = \sqrt{12} \sigma_{eq} = \sqrt{\frac{1}{T_{clk}} \sum_i B_i} \quad (2.74)$$

Although this method can achieve high resolution and precision, its main limitation is the high logic utilisation. In [158], the 16 delay line was used to accomplish a significant amount of improvement in the resolution and SSP. Therefore, when the amount of logic used is considered this method is not suitable for FPGA implementations, where large multi-channel operations are implemented. Also, due in no small amount of logic utilisation, no calibration was mentioned to be performed on the hardware.

2.5.5.4 Averaging TDC Technique

Another way to overcome the non-linearity problem of TDCs is processing the same trigger multiple times by using different delay lines. Averaging is primarily done by using a multi-hit scheme and generating a various number of codes and averaging their results to improve the DNL error [94]. This method can be implemented in two ways, one way is choosing the delay line to be much longer than the clock period, and thus, multiple logic transition for a single trigger can be generated in consecutive clock periods. The average of these codes is used as the final code of the TDC. This method is named as the Double Registration TDC [1, 139]. Alternatively, this can be implemented using multiple delay lines in parallel, as explained in [159]. With multiple delay lines the same trigger is routed to different delay line with the same number of bins at the same time and the clock edge the codes are captured simultaneously for each code. The resultant system was obtained by averaging these codes.

An illustration of the double registration averaging method can be seen in Figure 2.53.

An equation for measured T_{final} for a double registration TDC can be expressed as below, where C_1 , C_2 are two consecutive codes and T_{clk} is the clock period [1].

$$T_{final} = T_{clk} / (C_1 - C_2) \quad (2.75)$$

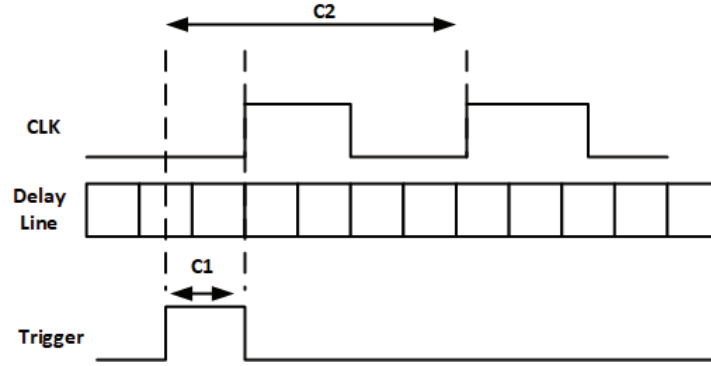


Figure 2.53: Illustration of the Double Registration TDC

An Illustration of a Multi-Chain Registration TDC can be seen in Figure 2.54.

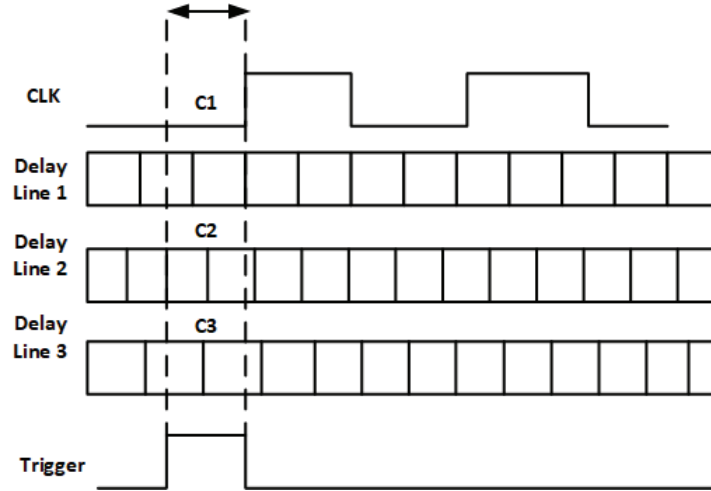


Figure 2.54: Illustration of the Multi-Chain Registration TDC

An equation for the T_{final} for the multi-chain registration TDC can be expressed as below, where three delay lines were used and C_1, C_2 and C_3 are the codes generated for averaging and N is the total number bins in each delay line.

$$T_{final} = \frac{(C_1 + C_2 + C_3)}{3 \times T_{clk}} \times N \quad (2.76)$$

The effects of averaging on the transfer function can be observed in Figure 2.54. Averaging creates a midpoint between two codes from different transfer functions. Thus, effectively averaging reduces the size of the large bins in the transfer function, which results in improved linearity.

Generally, averaging methods improve DNL errors with fast run-time. However, it is not a bin-by-bin calibration method, thus, it would be plausible to be used with a

bin-by-bin calibration method such as the code density testing. The main disadvantage of double registration method is the introduction of an additional clock period requirement for the quantisation, which results in an extra dead-time. Also, when the multi-chain method has used this results in further logic utilisation. By using the multi-chain averaging, in a Virtex-6 FPGA, 3 ps bins sizes with 6.5 ps SSP were achieved [159]. Averaging TDCs will be discussed in more depths in Chapter 4.

2.5.5.5 Sliding Scale Technique

Sliding Scale Technique is another TDC linearisation technique. This method aims to improve the linearity of the code by generating multiple codes using the same delay line then, averaging them for the final code [160, 1].

To improve the linearity, known delays are added to the input triggers, and different codes for each trigger is generated. It should be noted that triggers are the same signals and asynchronous to the system clock, but they all have various pre-set delays added to them. Thus, the difference between the START and STOP is kept unchanged, but, the code's location in the delay line moved around [1].

After multiple codes for the time difference between the START and STOP pulses are generated the added delays are digitally subtracted from the codes generated. Finally, codes are averaged to form the final code with improved linearity. A diagram for this implementation can be seen in Figure 2.55.

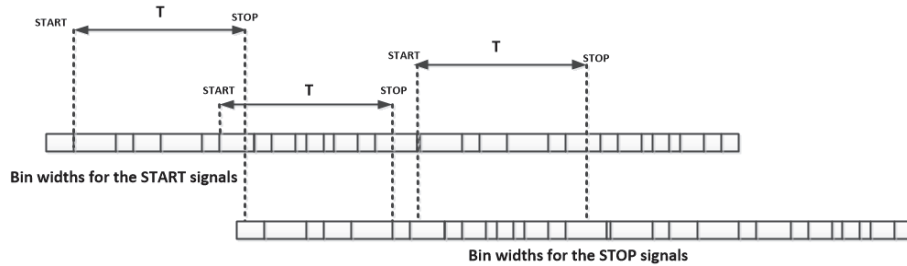


Figure 2.55: *Illustration of a Sliding Scale TDC [1]*

The main advantage of this method is improving the linearity of the delay line without implementing a bin-by-bin calibration. Although, as a TDC linearisation method it is a plausible approach, extending the size of the delay line in order to take multiple measurements for triggers introduces further dead-time. Also, since this method just improves the linearity of TDC, a bin-by-bin calibration is not achieved, and thus, a calibration method such as the code density testing would be useful to cooperate with this method. This method, achieved 17 ps SSP on an FPGA as it was described in [160].

2.5.6 Time-to-Digital Converter Summary

In this section, TDC schemes, TDC calibration and linearisation methods and measurement errors affecting TDCs were reviewed. The purpose of this review was finding a TDC scheme which would provide high precision, high count rate and multi-channels for a coincidence counting operation. Thus, implementations were compared according to this purpose. A compilation of all the reviewed TDC schemes and their performances in this thesis and author's publication in [1] can be found in Table 2.1. This table sorted in descending order of resolutions, and implementations' DNL, INL, SSP, channel numbers are compared.

First of all, in this review the technologies available for TDC implementations were compared. Due to the flexibility and cost of FPGA technology seem to be the best option for a coincidence counting instrument since it has shown to be providing high resolutions with flexibility and cost-effective compared to ASICs. Also, analogue designs suffer from the signal integrity and high dead-times which result in high costs with low count rates. Hence, FPGA technology is appeared to be the best option for the coincidence counting implementation. Later, various TDC schemes are compared in this section. The primary analogue methods were TAC-ADC, TE TDC and LPI TDC. These methods as it can be seen from Table 2.1 achieve high resolution such as 1 ps and 11.25 ps, but they typically only provide a limited number of channels and suffer from high dead-times and signal integrity problems. Also, they are not possible to implement on FPGAs.

Digital Methods discussed in this review can be listed as Algo TDCs, SA TDCs, DL TDCs, STDCs, Wave union TDCs, SERDES TDCs, VDL TDCs and Ring Oscillator TDCs (VCRO, GRO). As it can be seen from Table 2.1, Algo TDC, STDC, SA-TDC, Ring Oscillator TDC can achieve very high resolution respectively 0.61 ps, 0.61 ps and 0.7 ps. However, these methods are challenging to implement on FPGA, and in the literature high-resolution examples of these implementations are found in ASICs. Typically VDL, Wave Union Launchers and DL TDCs can be implemented on an FPGA. VDL implementations are challenging to implement on FPGAs due to the difficulty of implementing small variable delays. Wave Union Launchers can achieve very high resolutions as it can be seen in Table 2.1, but these implementations require multiple clock cycles to work, and thus, they have significantly large dead-times. The review pointed out that the optimal implementation for an FPGA based TDC as DL TDCs since they can operate at a single clock cycle dead-times, easy to implement on an FPGA and can achieve sub-10 ps resolution as it can be seen in Table 2.1. The thesis now continues with the coincidence counter review in the next section.

2.5. TIME-TO-DIGITAL CONVERTERS (TDCS)

Method	Tech-nology*	Reso-lution(ps)	DNL (LSB)	INL (LSB)	SSP (LSB)	Channels	Ref
Algo TDC	350	0.61	± 0.4	± 4.5	± 1.2	1	[136]
SA-TDC	350	0.61	N/A	N/A	N/A	1	[161]
STDC	130	0.7	± 1.4	± 2.4	N/A	1	[162]
Branching VCRO	65	0.85	± 0.27	± 2.94	N/A	1	[163]
TAC-ADC	N/A	1	N/A	± 10	1	1	[100]
GRO	130	1	N/A	N/A	N/A	1	[129]
SA-TDC	350	1.2	N/A	± 6.67	3	1	[127]
DL + TE	90	1.25	± 0.8	± 3	1	1	[164]
DL	FPGA	1.56	6	35	1.47	4	[107]
Algo TDC	350	2	N/A	± 1.25	± 0.15	1	[135]
STDC	130	4	N/A	N/A	N/A	1	[91]
Looped LPI-TDC	90	4.7	± 0.6	± 1.2	0.7	1	[122]
Looped LPI-TDC	90	4.7	± 0.5	± 1.0	N/A	1	[122]
DL	130	5	± 0.9	± 1.3	0.6	1	[110]
GRO + VDL	90	5.8	N/A	N/A	N/A	1	[134]
Wave Union A	FPGA	6	N/A	N/A	1	48	[165]
PS	130	6	± 15	N/A	1	1	[118]
GRO	130	6	N/A	N/A	N/A	1	[130]
LPI-TDC	180	7.84	N/A	N/A	N/A	1	[123]
PS + VDL	FPGA	8.5	0.36	0.91	29.9	264	[119]
SERDES	FPGA	10	± 0.4	N/A	15	64	[138]
VDL + DL	180	10	N/A	N/A	N/A	1	[166]
Wave Union B	FPGA	10	N/A	N/A	1	8	[139]
TAC-ADC	ECL	10	N/A	± 2	1.5	1	[98]
TE + DL	180	11.25	N/A	N/A	1.33	1	[167]
VCRO	350	12.2	N/A	± 0.41	0.66	1	[168]
Vernier SA-TDC	180	12.5	± 0.4	± 0.4	N/A	1	[125]
DL	FPGA	17	3.3	-2.99, +2.59	0.57	1	[105]
Looped PS	800	20	± 0.5	N/A	1	1	[117]
TE + SA-TDC	90	20	N/A	N/A	N/A	8	[169]
DL	90	21	± 0.7	± 0.7	5.58	1	[170]
VDL + VCRO	350	24	± 0.55	± 1.5	N/A	1	[171]
CRO	600	30	N/A	± 1.33	32	1	[168]
VDL	700	30	N/A	± 1.0	0.2	1	[172]
Hierarchical TDC	90	31.25	N/A	N/A	N/A	1	[173]
Dual-Slope TAC	800	32	N/A	± 0.16	0.94	1	[174]
DL	130	40	N/A	N/A	N/A	1	[175]
SA-TDC + VCRO	350	42	N/A	N/A	N/A	1	[100]
GRO	130	45	N/A	N/A	N/A	1	[176]
VCRO	130	50	± 0.5	± 2.4	N/A	1024	[177]
PS (2D Array)	800	50	N/A	± 3.5	N/A	1	[115]
VCRO	130	55	± 0.3	± 2.5	N/A	20480	[178]
Wave Union A	FPGA	60	± 1.17	± 1.08	0.42	1	[139]
VCRO	180	61	± 0.23	± 0.3	1	24	[179]
VDL + DL	FPGA	75	N/A	N/A	0.53	1	[180]
VCRO	65	80	N/A	N/A	N/A	1	[181]
DL + VCRO	350	97	± 0.09	± 1.89	N/A	32	[182]
TAC-ADC	500	312.5	± 0.2	± 0.3	0.32	1	[97]
VCRO	800	530	± 0.36	± 0.36	N/A	1	[183]
Multi-Phase Counter	180	780	± 1.9	-0.8, +0.5	N/A	1	[92]
SERDES	FPGA	1200	± 0.167	N/A	± 0.5	96	[137]

Table 2.1: Comparison of TDCs in the reviewed literature. N/A = Not Available. * CMOS ASICs in nm, FPGAs by series number [1].

2.6 Coincidence Counters

Coincidence counters are time correlations tools which are utilised for measuring the correlation between simultaneously happening time events. The time interval which the operation is performed is named as the coincidence window (T_w). The boundaries of this window is denoted by the upper bound (T_{gh}) and lower bound (T_{gl}). Detected coincidence pattern, which is formed across multiple channels, is defined as the address of a coincidence counter. A diagram for the coincidence counting can be seen in Figure 2.56.

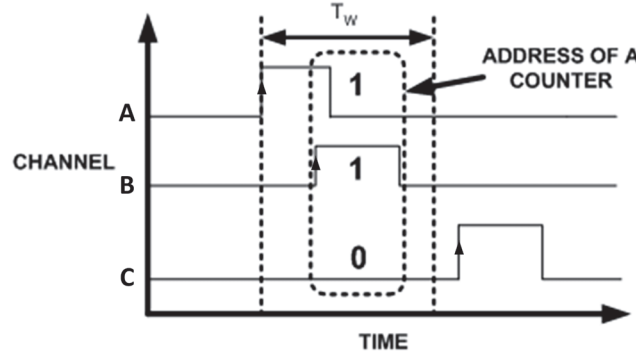


Figure 2.56: *Illustration of a coincidence address*

In this section, the various coincidence counting architectures available in the literature will be discussed from starting with the analogue coincidence detection and counting circuits.

2.6.1 Analogue Coincidence Detection and Counting Circuits

The Geiger-Muller (GM) Tubes are apparatus to detect the ionising radiation of particles such as α , γ and β . These tubes are typically cylinders filled with gasses such as Helium or Argon, and they accommodate two electrodes. When the radiation flows through the tube, it is ionised by the gas and detected by the electrodes [184]. Walter Bothe implemented the first-ever coincidence counter and published in 1930 [13]. This method was achieved by using two GM Tubes for detecting two simultaneous gamma and beta radiations in Cosmic Ray experiments, which resulted for him in winning a Nobel Prize in 1954 [185]. Later this method was improved by Bruno Rossi, and 3-fold coincidence detection circuit was implemented by using analogue circuit components and vacuum tubes. The Rossi's circuit utilised three vacuum tubes like transistor switches and one output tube whose output is affected by the change in the resistance when the 'switches' are closed. The input vacuum tubes are connected to the grids, where the negative current flows when an input is detected. When one or two inputs

2.6. COINCIDENCE COUNTERS

are detected, the current is not enough to change the resistance affecting the output valve. Still, when all the input valves are closed, this results in an interruption of the current flow and a sudden drop in resistance (R_d). Thus, the output of the vacuum tubes become 'closed switch', and due to the reduction in the R_d value, it outputs a positive pulse. The circuit diagram of this implementation can be seen in Figure 2.57. Rossi also reported that his implementation managed to support up to 10 channels. This implementation was also the first circuit implementation of a digital 'AND' gate.

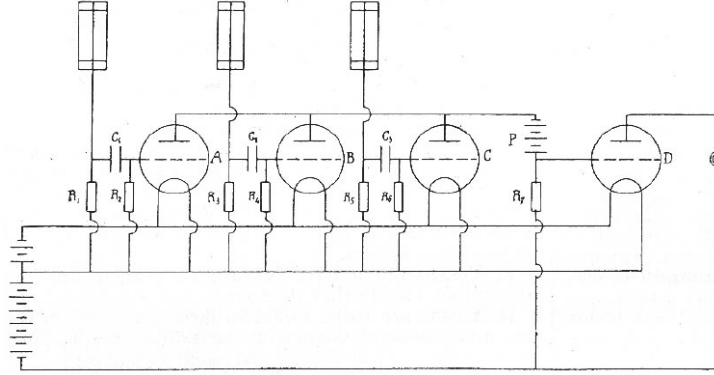


Figure 2.57: The original drawings of Rossi's coincidence circuit [13]

The idea of Rossi's coincidence circuit can be considered as an analogue coincidence detection circuit. Typically, analogue coincidence circuits use diodes as switches to determine when the input channels detect triggers at the same time. When either or both channels detect no input, the diode acts as a close switch and voltage stay at the zero [186]. However, if all the channels have an input than diodes act as open switches and conduct, which results in a rise of the output voltage V_c^+ . The required time for detecting simultaneous events is referred to as the resolving time in this kind of implementation. An illustration of this circuit can be seen in Figure 2.58.

Typically, these detection circuits are used in combination with counting systems, where events of coincidences are registered by using digital registration schemes such as ADCs alongside with the individually digitised triggers for each channel. This type of implementations are referred to as Double Conversion Coincidence counters, and thus, each channel's events are registered along with the coincidences. Since these implementations were initially designed for systems using obsolete technologies such as tapes and punch cards, they accelerate the operation by registering coincidences separately to the memory. Thus, the single events for each channel and coincidences between channels are stored independently. For instance, in radiation measurements, this information is used for measuring the radiation levels of α , β and γ rays and their coincidence radiations [186].

These methods are usually found in older publications in the literature, such

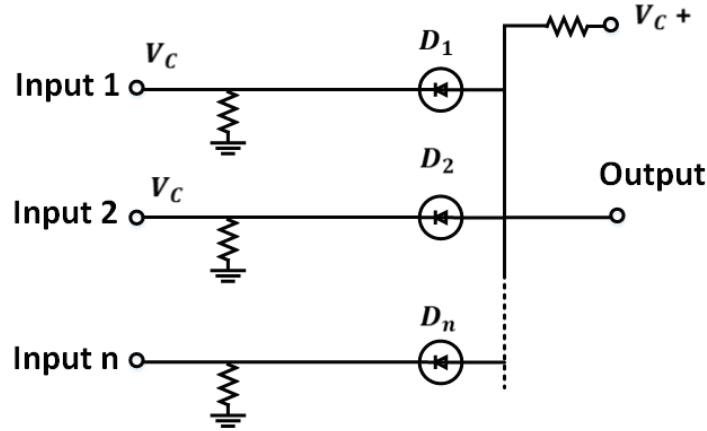


Figure 2.58: Illustration of an analogue coincidence circuit

as [187] (1962), where, three-channel coincidence circuit for scintillation counters implemented, the resolving time was reported to be 1.8 ns and dead-time with 300 ns. Primarily, the analogue implementations are used for implementing 'AND' gates. However, with developments in digital electronics, this implementation has become obsolete and replaced by their modern counterparts due to the simplicity and flexibility of digital implementations.

2.6.2 Pulse Height Analysing Coincidence Counter

One way to implement a coincidence counter is utilising the overlapping energies of the coincident photon events (quasicoincident photons) since they are coinciding. This method uses the photon detectors' energy discrimination capabilities for analysing pulse heights since the Photon Counting X-Ray Detectors (PCXD) observe the overlapping events and see coincidences as superposition of energy peaks. This method operates by using energy thresholds to determine the coincidences because overlapping events' combined energy should exceed the threshold value [188].

In the implementation described in [188], an ASIC circuit has been utilised to implement the energy threshold check and count the coincidences. The ASIC module used for the coincidence counting accommodates a pre-amp circuit, a pulse shaper, two DACs, two comparators and two counters. DACs are used for setting the threshold values for the comparators, and each counter counts up separately when the input exceeds their threshold energies. Since each counter has different energy levels, to find the coincidences in an energy window, the counts of these counters are subtracted from each other. An illustration of this scheme can be seen in Figure 2.59 and its waveforms in Figure 2.60.

This method is conceptually simple and reported to provide 50 MCPS per channel, the precision of the technique is reported as in the range of energy window, where

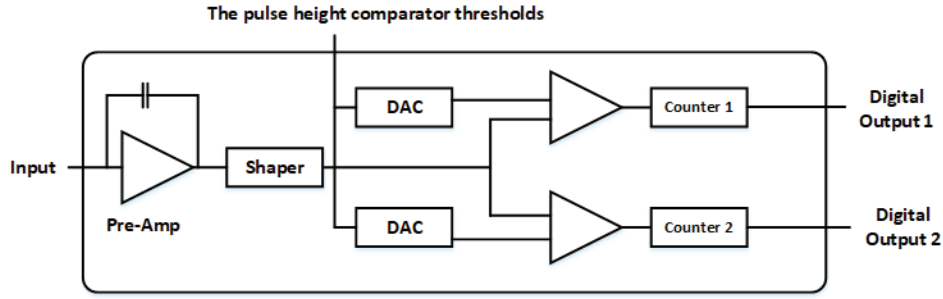


Figure 2.59: Illustration of height analyser coincidence detection block

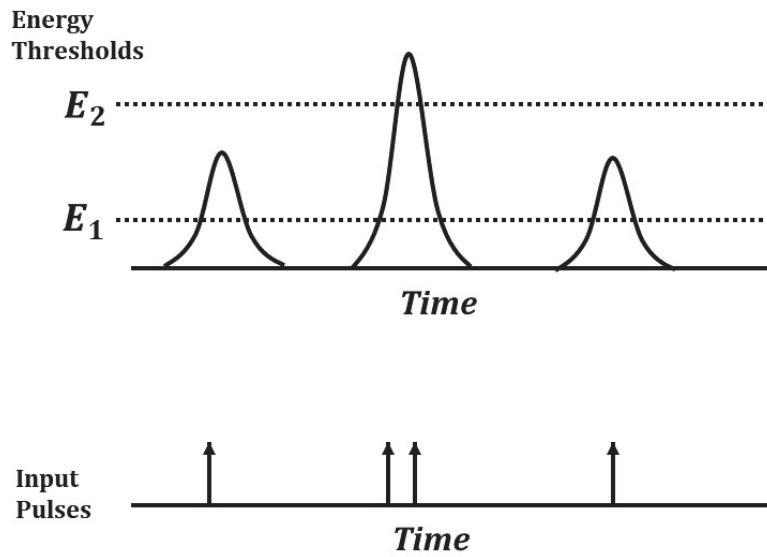


Figure 2.60: Illustration of an energy levels caused by pile-up

the resolving time was recorded was 81.2 ns and the smallest energy window was 17 keV. Since this method was designed for measuring the energy of photons with PCXD's rather than the time of arrival, and so, it is not ideal for quantum photonics applications, where the time of arrival is interested, and SNSPD is typically used with pulse lasers.

2.6.3 Height-To-Width Converter Coincidence Counter

Another coincidence counting implementation in the literature is using height to width converters and semi-period measurement technique as it was described in [189]. This method is developed for coincidence counting measurements used in nuclear radiation experiments, where the coincidence events and their radiation energies are of interest. The width conversion is applied by determining the decaying time of the trigger, and

hence, how long the input's voltage value stayed above the threshold V_t is recorded. The width of the output defines the energy of the radiation. V_{out} signal is generated for the time the signal's decaying voltage stayed over the threshold voltage [11].

Generated signal V_{out} goes under a semi-period measurement, where a height-to-width converter generates values for the pulse widths. The number of the clock edges are passed during the width is counted. This counting includes the width between both rising and falling edges. The width values framed by rising edges named as the leading edge timing. The width values for the falling edges are named as the falling edge timing [11].

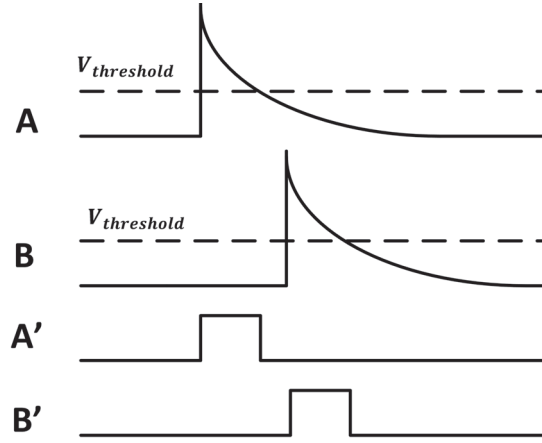


Figure 2.61: Generation of the trigger based on the input decay

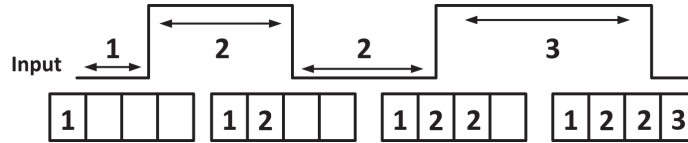


Figure 2.62: The semi-period measurement [11]

These timings are stored in a time-sequenced manner in an array. The timing values of the rising edge are stored at even indices, and falling edges are stored at odd index. The leading edge timings should be less than the coincidence window to be coincidences. The width of the pulse determines the energy of the coincidence event. Later this information is used for energy spectrum analysis of radiation events.

This type of coincidence counter is not ideal for achieving the best possible precision since the semi-period measurement is more application-oriented than the precision. Therefore, with this method, only 12.5 ns time resolution was provided. However, it is still a good example of how time stamping of events offer flexibility in terms of application needs and can be used to obtain different properties such as energy

spectrum in nuclear chemistry.

2.6.4 Combinatorial Logic Based Coincidence Counters

Coincidence counters are commonly implemented using combinatorial logic circuits. The simplest way to achieve a coincidence counter would be using an 'AND' gate. In an 'AND' gate based implementation, the gate input all the channels and outputs logical high when all the inputs are high. In this method, the width of the input pulse that can be still detectable by the system determines the coincidence window size. Thus, if the coincidence detected forms a pulse which is too short for the system register it will be missed, i.e shorter then the flip-flops critical times. Also, system's capability of generating short pulses affect the coincidence window size since gates which can output with smaller propagation delays can be used for generating shorter pulses. Typically, 'OR' gates are utilised for choosing inputs for the 'AND' gate for multi-fold operations when a specific pattern is interested. A commercial example of an AND gate based coincidence counter is ORTEC Co4020 Quad 4-Input, which provides 10 ns window size [190]. 'AND' gate based implementations are common in the literature, one of the examples is [191], where approximately 21.5 ns coincidence window was achieved on Cypress Programmable System-on-Chip (PSoC). Probably, one of the best-achieved coincidence performance using a AND gate in the literature is [192], where the 1 ns window was achieved by using Positive Emitter-Coupled Logic (PECL) 'AND' gate and a pulse shaper. Also, another pulse shaping circuit using 'AND' gate is [36], where 7-10 ns coincidence windows were achieved and 'OR' gates are used as a part of the channel selection process.

Illustration of AND gate based TDC can be seen in Figure 2.63.

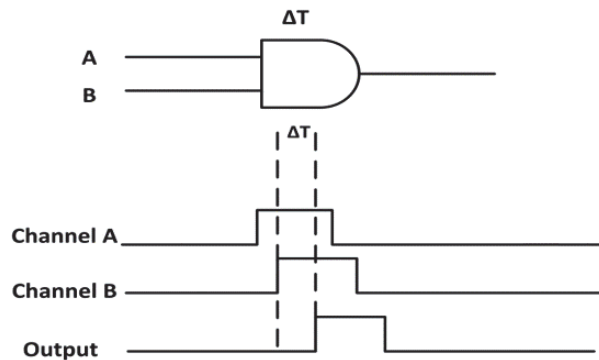


Figure 2.63: Waveforms of and gate coincidence counter

Another combinatorial logic method to implement a coincidence counter is using pulse shaping circuits. These circuits aim to reduce the widths of input pulses to improve the performance of the operations. In this method, the signal path of the

CHAPTER 2. RESEARCH REVIEW

input gets duplicated, and one of the paths gets inverted. This inversion introduces an additional delay to the path, and this results in a logical high for a shorter time interval, which acts as a pulse shaper [12].

An illustration of a pulse shaping circuit can be seen in Figure 2.64, where two pulse shaping 'AND' gates input A and B with the inversions of these two signals which are A' and B' . By using inverters propagation delay, the output of the inverter and the signal's itself AND for the pulse shaping. Finally, outputs of pulse shaping circuits of A and B signals are input an 'AND' gate for the coincidence detection. The design explained in [193], can be an example of this architecture. The pulse shaping method effectively reduced the size of the coincidence window, and it was reduced from 25 ns to 10 ns. Also, [36] and [192] also used pulse shaping to improve the resolution of the coincidence detection.

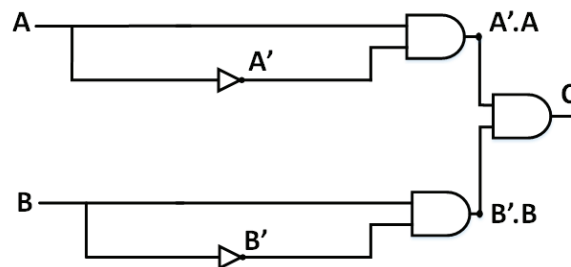


Figure 2.64: *Illustration of a pulse-shaping coincidence counter*

The waveforms for the pulse shaping coincidence counter can be seen Figure 2.65,

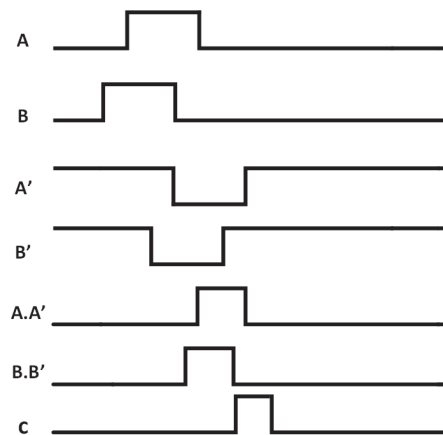


Figure 2.65: *Waveforms of a pulse-shaping coincidence counter*

Typically, AND gate methods are needed to be registered with the clock edge to be used in digital systems. Therefore, if the output signal is not wide enough to be detected by the system's clock edge, it will be missed by the system. The simplest way

2.6. COINCIDENCE COUNTERS

to solve this is by using an edge detection circuit to implement a coincidence counter. In this method, inputs are synchronised with the edge of the clock by using a D-type flip-flop. The output of the flip-flop goes through an inverter. Outputs of flip-flops feed an 'AND' gate, which effectively detect coincidences with the rising edge of the clock [194]. With this method in [194], 8 channels 4-12 ns window sizes were achieved with 800 KCPS. Also in [195], by using D-type flip-flops and variable phase differences between inputs and the clock is used to implement a coincidence detector. Variable delays essentially act as a variable window size and the flip-flop's gate time is used as the minimum window size, which was 1 μ s.

An illustration of this method can be seen in Figure 2.66, where A and B inputs are synchronised, two D-type flips and their inversion with themselves are input an AND gate for the coincidence detection. Also, the waveform for this method can be seen in Figure 2.67.

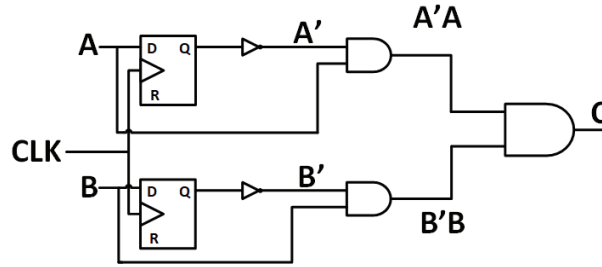


Figure 2.66: *Illustration of an edge detection coincidence counter*

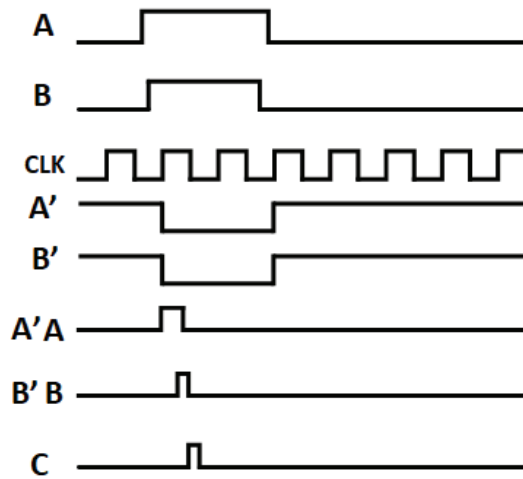


Figure 2.67: *Waveforms of a edge detection coincidence counter*

However, when coincidences occur right after the clock edge, but far shorter than the next clock edge, this method will not be able to detect coincidences. Thus, the

Asynchronous Latch Coincidence Counter has been developed to solve this issue [155]. In this method, as it can be seen in Figure 2.68 and Figure 2.69, there are monostables (L1, L2, L3, L4) which controls the coincidence window width and FIFO enable logic. L1, L4 are rising sensitive, while L2 and L4 are falling edge sensitive monostables. The input signals A,B and C feed an 'OR' gate to enable the monostable logic with the first rising edge of the input while N-bit Asynchronous latch is used for capturing the coincidence pattern. With the first rising edge of the signals, L1 is activated, and with the falling edge of the L1's output (L1'), L2 is enabled. L2's output is named as L2'. The total time of L1 and L2 monostable are activated act as a coincidence window. L3 and L4 monostable provide sufficient time for the captured coincidence pattern to be registered by the FIFO. The FIFO outputs the coincidence counting logic with the output of L4 (L4'). This method has previously implemented in University of Bristol Photonics Group as it was described in [155] and 1.17 ns window size was achieved. The biggest issue with this method is that the configurable channel delays were difficult to achieve since reconfiguration of entire FPGA logic was required when delays were needed to be updated. Thus, this implementation failed to provide flexible coincidence counting operation.

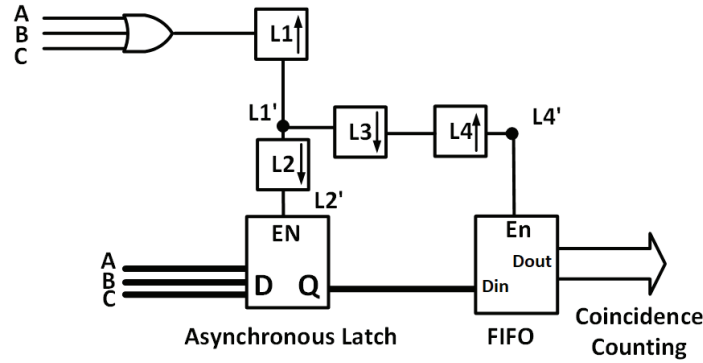


Figure 2.68: *Asynchronous latch coincidence counter*

2.6.5 TDC based Coincidence Counters

One of the popular ways to implement a coincidence counter is using time tags generated by TDCs. TDCs are used for timestamp generation in different applications. These timestamps make high resolution and flexible coincidence counting possible. With many TDC based coincidence counters, pico-second range coincidence window sizes are achieved and being used in areas such as photonics, medical imaging and nuclear science [3, 31, 196].

The idea behind using timestamps is by comparing the time tags from different channels reveal the time difference between the trigger events. Timestamps consist of

and commercial products. One of the best examples of TDC based coincidence counter system in the literature can be the ToF PET coincidence counting system, which was described in [31]. This implementation was based on a 12-channel TDC board, which transfers time tags to an FPGA to be analysed for coincidence counting. The resolution of the system was 68.3 ps, and the system could process a total of 72 MCPS and used for the PET image reconstruction. Another example of a TDC based coincidence counter was implemented for the animal PET scanner as it was described in [33]. This implementation used an ADC based TDC implemented in an FPGA, where generated tags are transferred to the DSP module to analyse time tags for coincidence counting. The system had 8 operational channels, 0.7 ns resolution while providing 32 MCPS performance. Also, for Boson sampling, a large scale TDC based coincidence counter was implemented as it was described in [32]. This implementation had an impressively large scale of 32 channels, and however, the implementation was providing low resolution with 390 ps with 500 KCPS due to a large FIFO involved in the process.

In the commercial side, as it was previously mentioned, PicoQuants and ID Quantiques products can be an example of TDC based coincidence counters. When the performance of the top of the line products are compared, PicoQuants HydraHarp400 achieved 12.5 MCPS per channel with total 96 MCPS across 8 channels with a bin width 1 ps with 8.5 ps SSP [20], ID Quantiques ID900 which had 4 channels with 25 MCPS per channel and the total of 100 MCPS with 8 ps SSP [30].

2.6.5.1 Rolling Window Method

One way to implement a TDC based coincidence counter is by using a rolling window method. In this method, the coincidence window is set to the beginning of the first time tag, and by rolling the window half of the window width, the coincidences can be checked. This process starts with a generation of time tags upon detecting trigger events. Time tags are generated in parallel by channels and tags are sorted based on their time of occurrence. Typically, a buffer is used for storing these time tags in a time-sequential manner [12, 11, 155].

The algorithm searches inside the buffer to find coincidences. The coincidence window T_w is set to the first window, and the coincidence pattern is checked. Once coincidences are tested, the window is rolled forward by half of its width ($T_w/2$). This coincidence rolling process is continued until there is no tag left inside the buffer. In Figure 2.71, an illustration of a method can be seen.

This coincidence detection is done by setting the coincidence window boundaries to the lower boundary T_{gl} is set to closest tag and the upper boundary T_{gh} is set to $T_{gl} + T_w$. Then, the tags are checked for whether they lie within the coincidence window, and the coincidence pattern's bits are set based on events' channels. As a result of this

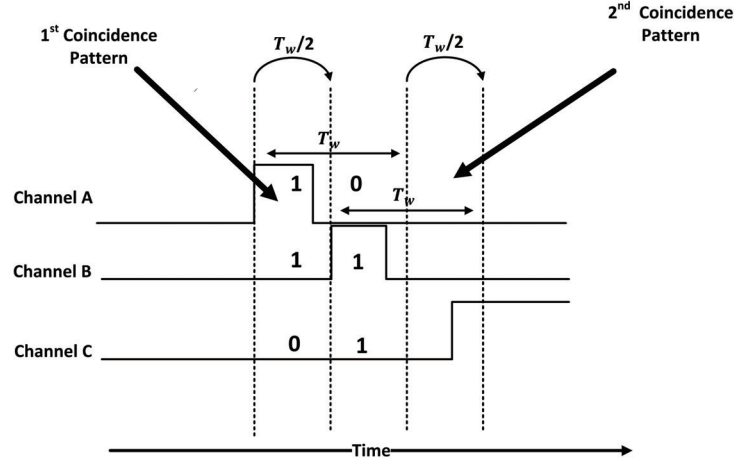


Figure 2.71: *Illustration of a rolling window method*

process, the coincidence patterns are formed. Once a coincidence pattern is formed, the window is shifted by half of its length [12, 11, 155].

This method is a straightforward and easy way of implementing a coincidence counter since only comparator and iterator logics are needed to traverse inside the buffer. Therefore, this can be a plausible coincidence counting implementation for FPGAs with limited resources. However, the operation time is relatively long since iterating the half the window size could be very exhaustive for long-distance measurements when there are vast distances between adjacent tags. Nevertheless, for specific applications, where the window sizes and ranges were well balanced, this approach can be used. The first mention of this approach can be seen in [155] and later in [12], approximately with 120 ps window managed to capture all coincidence counters generated in an 8 channel system.

2.6.5.2 Forward And Backward Looking Tag Differences Methods

Another similar approach to the rolling window method is forward and backward looking tag difference methods. The main difference between these two methods and the rolling window method is instead of moving the coincidence window, the time difference between tags are calculated, and these time differences are compared with the coincidence window. Being backwards or forward implies the direction of the iteration inside the buffer of time-sequenced time tags. However, the system can process time tags faster with a forward looking tag difference method, and it could be used for implementing a coincidence counter without using a tag storing buffer as it was described in [7, 3]. On the other hand, the backward looking method must have a buffer to check the tags backwardly.

The forward search subtracts adjacent tags from each other as they occur and

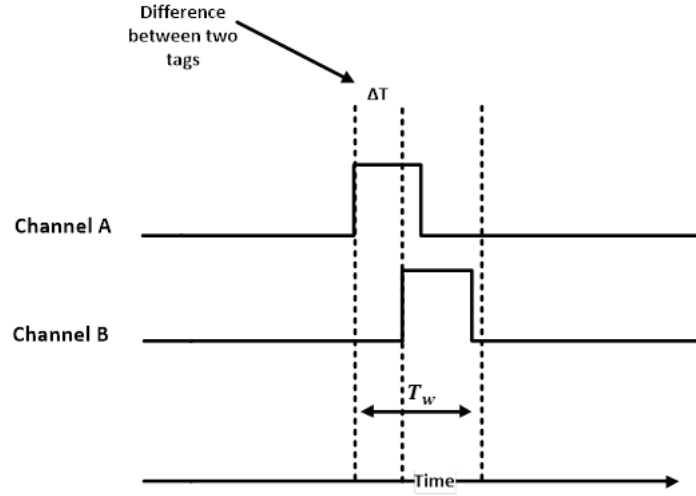


Figure 2.72: *Illustration of a forward looking tag difference method*

by merely comparing time differences (ΔT) with the coincidence window, coincidence patterns are formed. The algorithm starts with subtracting the first tag from the second tag and checks whether they are building a coincidence. This process is repeated from the first tag's perspective until tags are appeared to be outside of the T_w . Then the process is continued with the second tag. Thus, in the worst case, the run-time of this method would be the factorial of the entire buffer size (n) ($O(n!)$). However, this method can be improved by using concurrent multi-stop TDCs, where the time differences between the START and STOPs are continuously calculated. In the worst case, the run-time of this new method would be $O(n)$ where n is the time between two START signals. Therefore, the forward looking approach seems to be advantageous for real-time operation. An Illustration of a forward looking time difference method can be seen in Figure 2.72.

Another disadvantage of using a buffer to implement this method is the read and write pointer collisions. During the operation of coincidence counting, searching through the buffer frequently results in a clash between the read and write pointers, since the coincidence detecting is likely to be faster than the tag generation. This might slow down the operation because of corrupted data.

Alternatively, when the real-time operation is not concerned, the backward looking tag difference method can be used. In this method, reading tags starts from the bottom of the RAM while the writing begins from the other end. Therefore, only once the write and read pointers clash while traversing through the RAM. However, this method introduces long delays since the backward counting implies that coincidence counting will be towards to the most recent to the least current tag and there will be some wait required until enough measurements are stored in the buffer. An example of backward

looking tag difference coincidence counters can be seen in [11], which was used in the earlier iterations of the coincidence counting system and the recap of the results achieved from these implementations can be found in Appendix 8.3.3.

2.6.5.3 Multi-Channel Labelling Coincidence Counter

One of the main obstacles faced when implementing a coincidence counter-based TDC is transferring time tags to the coincidence counter with a high data-rate. Primarily when the design is implemented for multiple channels, the data-rate of the TDC is significantly hindered by the requirement of the data serialisation. In order to avoid the data serialisation, the coincidence counting operation can also be paralleled for multiple channels, and by using the non-buffered version of the forward looking tag difference method, a real-time operation can be achieved. However, having parallel coincidence counters, likely to cause counting the same coincidences multiple times by different channels, and the channel labelling method has been developed to cover this issue.

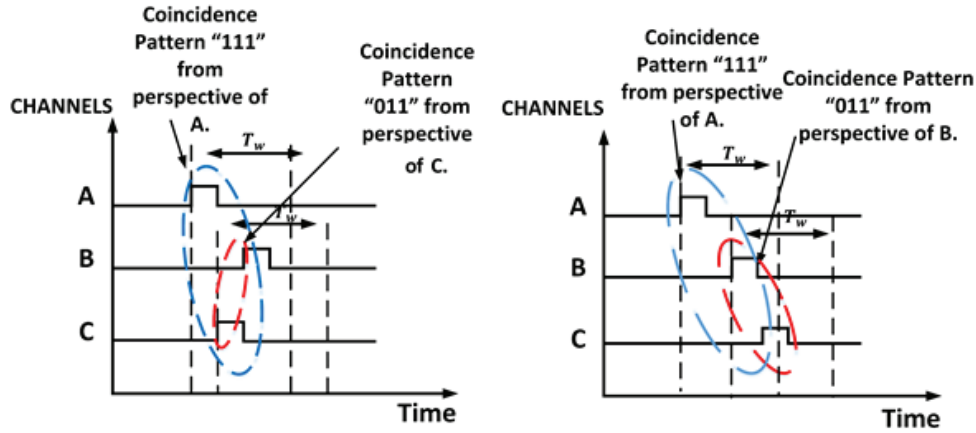


Figure 2.73: *The coincidences detected from different channel perspectives*

Channel labelling coincidence counter was developed for the integration of coincidence counter and TDC into the same FPGA chip [3]. Since time tags are generated and processed in parallel, channels do not know whether coincidences are detected previously by another channel. Channel labelling method labels each coincidence data based on the channel it is detected. After the coincidence counting is completed, in the post-processing step, uncounted coincidences are counted, and duplicated coincidences are discarded.

The post-processing starts with detecting and counting coincidences in parallel from separate channels. Each channel accommodates a forward looking tag difference method based coincidence counting modules, which operate from its own perspective.

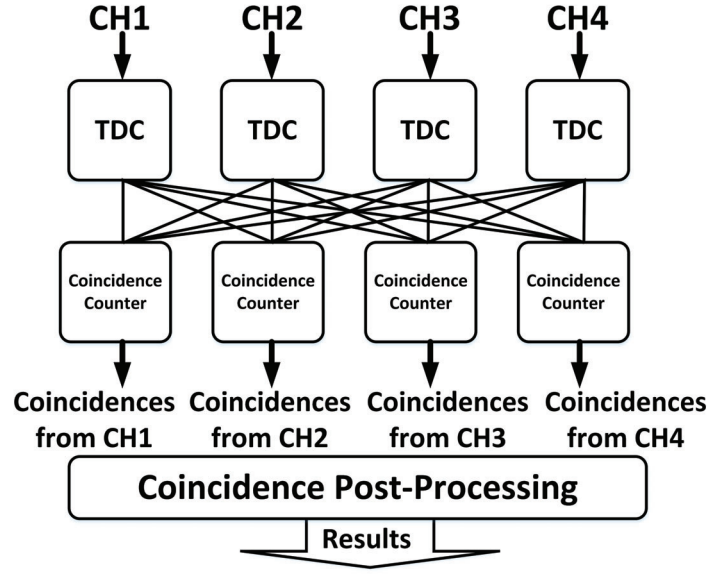


Figure 2.74: Operational block diagram of multi-channel labeling coincidence counter and the tag exchange between channels

During the coincidence counting operation, coincidence detectors aim to find the largest possible coincidence fold based on the channel's perspective, and so, the sub-folds initially will be ignored. Once the integration time of the coincidence counter is reached, each channel outputs the counter values for its coincidence addresses. In the post-processing step, the sub-folds from larger coincidence patterns are found, and duplicate counts are discarded. This scheme provides the best possible real-time coincidence counting operation since the method does not require any tag serialisation or buffering, because of this reason, it was used in the coincidence counting system that has been proposed in this thesis. The results obtained with this method will be discussed in the later chapters. The details of this method will be discussed in Chapter 5 and for the results Chapter 7.

The main limitation of this design is the need for larger RAM blocks and as the number of channels increases the memory needs of the system increases exponentially. Therefore, the memory requirements of this scheme can be expressed as below.

$$M = N \times W \times (2^N - 1) \quad (2.77)$$

Where M is the total bit size of Block RAM (BRAM) required for the implementation, N is the number of channels, and W is the bit size of each tag. Therefore, for 8-channel coincidence counter would need around 65 kbit space for 32-bit time tags. The large amount of memory usage would yield more RAM block usage and in smaller FPGAs this would typically result in difficulty of routing of the logic and meeting the

timing constraints. Therefore, for large number of channels are expected to be less cost efficient and more resourceful FPGAs such as Xilinx UltraScale line can be considered [198] where total of 75 Mega Byte (MB) of RAM available.

2.6.6 Superconducting Single Flux Quantum Coincidence Circuits

In recent years with developments in cryogenic, the superconducting logic has become an option to implement a coincidence detection circuit as it was described in [199]. Single Flux Quantum (SFQ) utilises superconducting circuits named as Josephson Transmission Lines (JTL), which operate at very low temperatures typically below 10 K. JTL provides a very high time resolution due to its extremely conductive nature, and in [199], 0.01 ps resolution was reported.

The implementation is similar to the time tag-based coincidence counters, but JTL based delay line is used for the time quantisation. Signals detected by SNPDs are denoted as the START and STOP signals, and they input Magnetic Coupled DC/SFQ (MC-DC/SFQ) converters. After the signals are converted to the SFQ, the START and STOP signals propagate through JTLs (JTL_1 JTL_2). The comparator circuit determines the time difference between the START and STOP based on the discriminator. The discriminator was reported initially in [200], where two separate JTLs with different variable delays are raced with each other like in Vernier Method TDC, and the START SFQ pulse is captured by a flip-flop with the STOP SFQ signal. Thus, the START SFQ only appears at the output if it is happening before the STOP. The optical delays affecting JTLs can be varied between 0.01 and 400 ps; these optical delays mainly determine the coincidence window. A circuit diagram for this implementation can be seen in Figure 2.75.

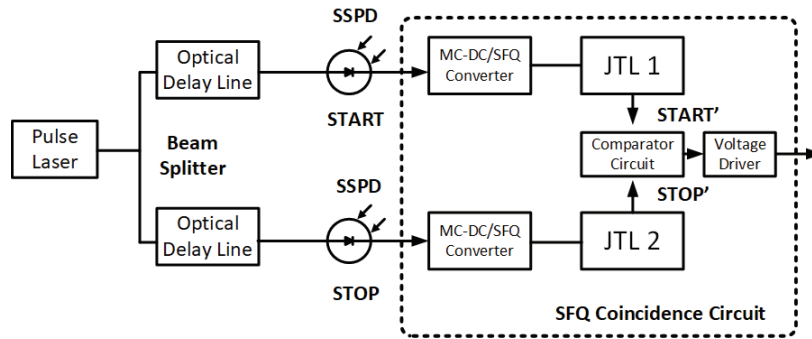


Figure 2.75: The system diagram of SFQ coincidence circuit

The main advantage of this method is a precise timing jitter provided by the SFQ. This method was reported to have a timing jitter of 1.1 ps FWHM, which was extremely low in comparison with any other digital timing circuits. Also, 800 ps coincidence window used in [199], and but, this value could be adjusted by changing

the variable delays affecting the optical delay lines in multiples of 0.01 ps. However, cryogenics is relatively new, complicated and expensive technology to use for most of the applications. Due to the need for low temperatures, it is costly and challenging to integrate into every application, and the high count-rate or multi-channel performance of these devices has not been reported yet.

2.7 Summary

In this chapter, the fundamental concepts to understand the work presented in this thesis were covered. The concepts covered in this chapter included conventional LiDAR methods, an introduction to the quantum information, TDC technologies, TDC techniques and coincidence counting methods.

The research review pointed out that, a pulse LiDAR technique can be used effectively used to measure the ToF of photons in quantum photonics experiments with using TDCs. Additionally, the Multi-STOP LiDAR method was shown to be effectively used for implementing a multi-channel coincidence counting method. For achieving a timing instrument, the FPGA technology is possibly the preferable approach compared to the analogue and ASIC methods, when the flexibility, costs and performances were considered. By using a tapped delay line method, a high precision FPGA based TDC with around 10 ps SSP can be obtained while keeping the logic utilisation and dead-time in moderation compared to other methods. However, in order to keep the precision high, the non-linearity of the TDC has to be tackled, which is mainly caused by FPGA's internal structure. The code density calibration technique can be used for a bin-to-bin calibration. Compared to the direct calibration technique, this is a less exhaustive and straightforward way to implement a TDC calibration, and also, the code density calibration was previously proved to be working on-the-fly. Additionally, to further improve the SSP and achieve sub 10 ps precision, a linearisation scheme can be utilised. Typically, these schemes are sliding scale technique or averaging based methods. However, the averaging based methods are probably the most straightforward way and common ways to achieve an improvement in the SSP and quantisation errors. Therefore, a scheme which uses an averaging method without introducing a dead-time or an extra logic utilisation could be beneficial for high count-rate operation in complex quantum photonics experiments. A proposed averaging based linearisation scheme with the TDC architecture will be covered in Chapter 4.

The research review also analysed, the coincidence counting implementations in the literature. The modern coincidence counters which are used in many different applications typically, either use 'AND' gates or Time-tagging techniques to achieve coincidence counting. However, generally, time tagging based methods provide flexibility with precision, since time tags can make very high precision while provides variable digital delays and coincidence window sizes due to their digital nature. On the other

hand, 'AND' gate based methods are less-flexible, and typically it is problematic to change the channel delays affecting the operations or window sizes. Also, their precision can not be easily improved since it is limited by the propagation delay affecting the gate. Therefore, TDC based coincidence counting is more beneficial for the proposed architecture. Among all the TDC based coincidence detection schemes tested for this research, the forward looking tag difference method is shown to be the ideal method for a fast coincidence counting operation since it can be used without a tag buffer. Also, the multi-channel labelling coincidence counting scheme should provide the fastest multi-channel operation for a complex multi-channel quantum photonics applications due to its channel concurrency. The proposed coincidence counting architecture will be discussed in Chapter 5 and the legacy of the proposed scheme will be addressed in Chapter 3.

After a sufficient amount of background provided and methods for implementing TDCs and coincidence counters were discussed, the thesis will now continue with the legacy coincidence counting systems designed by the author.

CHAPTER 3

Legacy Coincidence Counters

3.1 Introduction

An efficient scheme to provide a high count-rate precise multi-coincidence counting system has been developed in a few iterations. Probably the ancestor of this project could be considered as the coincidence counter designed by Nock. R as it was explained in [155]. That implementation was using an asynchronous latch based coincidence counter design which was implemented in the FPGA fabric for a fast coincidence counting operation. However, the flexibility of this work was limited by the difficulty of setting variable channel delays. Later in 2015, a software-based time tagging coincidence counter was implemented by the author [12]. Due to the usage of software, the real-time operation of this implementation was limited, and however, the forward looking tag difference method was tested successfully. In 2016 [11], the software-based coincidence counter has been migrated to a Spartan 6 FPGA, where the time tags were stored inside the BRAM block for the backward looking tag difference method. On the other hand, using a RAM for the tag buffering was mitigating the real-time operation, and errors were observed, which will be discussed later in this chapter. After the first attempt on implementing a real-time coincidence counter in an FPGA, to improve the real-time operation of the coincidence counter, the implementation was parallelised by using multi-tag correlators which were described in [7]. However, this method was using a First-in First-out (FIFO) to feed the coincidence counter. Thus, the real-time was limited by the data serialisation performance. Finally, the complete concurrent integration of a TDC and coincidence counter achieved where the operation was fully parallel and real-time, which will be discussed in Chapter 5.

In this chapter, the legacy iterations of the coincidence counter implementations will be discussed, with starting from the software-based coincidence counting implementation.

3.2 Software based Real-Time Coincidence Counting System

The first attempt on implementing a TDC based coincidence counting was made as an MEng Research Project by the author in 2015 as it was described in [11]. The first coincidence counter implementation was using the TDC instrument designed in Photonics group in the University of Bristol [155]. The TDC was designed on a Spartan 6 FPGA where a carry chain based TDC was implemented using CARRY4 blocks, and the coincidence counter was implemented with software which processed time tags every 50 ms. This implementation was aimed to replace the coincidence counter previously implemented for timing instrument designed within the Photonics group, which was based on an asynchronous latch and where the smallest window achievable was around 1 ns [155]. With this implementation, a preciser coincidence counting operation was aimed to be designed. The rolling window and the forward looking tag difference methods were tested for the first time in this implementation.

The designed system's overview can be seen in Figure 3.1.

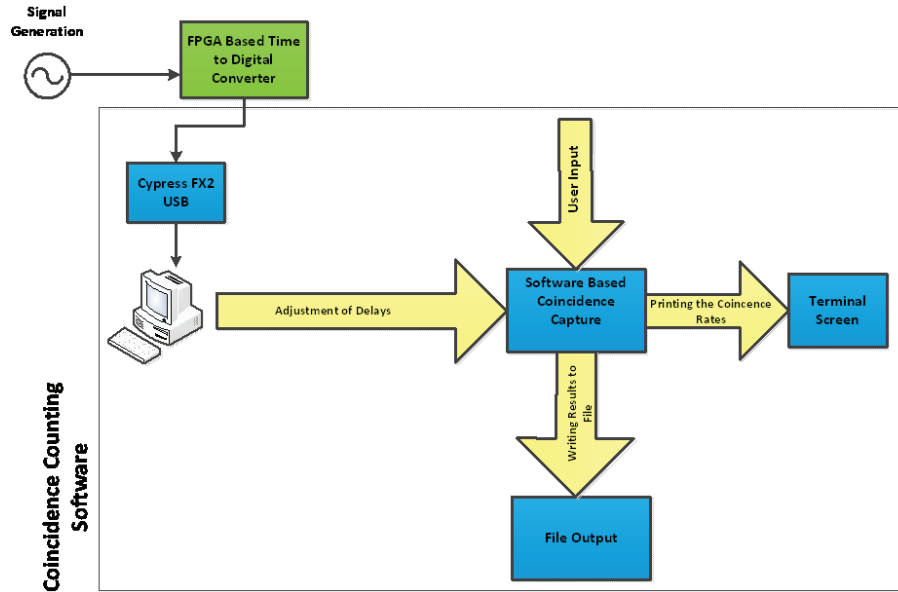


Figure 3.1: *The TDC based coincidence counter software [12]*

3.2. SOFTWARE BASED REAL-TIME COINCIDENCE COUNTING SYSTEM

The system features were as follows :

- 1.87 ps LSB resolution and adjustable window size.
- Multi-channel support up to 8 channels.
- 50 ms real-time operation.
- Up to 8-fold coincidence detection.
- Adjustable delays in each channel.
- Terminal and file-based output of results.

3.2.1 Software Implementation

In this implementation, the rolling window and forward looking tag difference methods were tested, which were explained in Chapter 2. These implementations typically designed to provide several functions as they are listed below [12]:

- Interfacing the FPGA to receive time tags' as bulk data.
- Multi-channel operation support up to 8 channels.
- Storing time tags to a data structure for processing.
- Running the rolling window and forward looking tag differencing algorithms on the data structure.
- Outputting the results.
- De-allocating the allocated space for time tags for the next bulk of tags.

The implementation was written in C programming language, and the libusb-win32 library was used for implementing the USB interface.

As the data structure, a linked list was used due to the need for dynamic memory allocation. The linked list had sequentially allocated nodes with pointers to the next node, which were used for traversing through the stored time tags. The linked list's sequential structure was useful since time tags were generated sequentially. The coincidence software output the counted coincidences every 50 ms and were requesting another bulk of time tags from the time tagging hardware. This 50 ms was kept tracked by reading the time tag values. It was also essential to de-allocate the memory after each run since otherwise, the computer would run out of RAM space.

CHAPTER 3. LEGACY COINCIDENCE COUNTERS

Rolling Window Implementation : The steps of the rolling window implementation could be summarised as below [12].

1. The algorithm starts with downloading a bulk of time tags from the FPGA.
2. The downloaded tags are stored into a linked list data structure in a time-sequenced manner for coincidence counting operation.
3. The program continues running the coincidence counting until the 50 ms time is elapsed, then new bulk of data is downloaded for processing.
4. The 50 ms integration time is tracked by checking the tag difference between the first and the last tag difference.
5. The algorithm first places the smallest tag to the head of the linked list then, starts iterating and sets the lower bound (T_{gl} to the first tag and upper bound(T_{gh}) was set to the $T_{gh} = T_w + T_{gl}$.
6. The algorithm checks time tags until there is no tag left and the next received tag is out-of-border.
7. Once a tag which is greater than the window is found, the coincidence window is rolled by half of the window width ($T_{clk}/2$).
8. This process continues for all the tags in the linked list; then results are displayed and written to a file.
9. After the process is completed, the linked list's content is deleted, and the system goes back to the first step.

The Forward Looking Tag Difference Implementation : The forward looking tag difference method algorithm was following steps listed below [12]:

1. The method starts with downloading the time tags from the FPGA via USB cable.
2. Like in the rolling window method, the downloaded tags are stored into a linked list in time-sequential manner where they will be checked for coincidences.
3. Algorithm finds the smallest tag and sets the window boundaries according to it.
4. Then algorithm starts checking the time difference between the current tag and the adjacent. If the time difference is less than the coincidence window size, the coincidence pattern corresponding the channel of the tags are found. Coincidences are checked until the time difference between tags are greater than the window size.

3.3. BACKWARD LOOKING TAG DIFFERENCE FPGA BASED REAL-TIME COINCIDENCE COUNTER

5. Once the time difference is greater than the window size. The window is set to the next lowest tag.
6. When the 50 ms integration time is completed, results are written to a file and displayed on the screen. Then the linked list is reset for the next bulk of data, and the algorithm continues with the first step.

3.2.2 Summary

In the University of Bristol Photonics Group, the first attempt on implementing a coincidence counter using time tags was made in this implementation. This design used the forward looking tag difference and rolling window methods for coincidence counting. These coincidence counting schemes were implemented on software where the time tags were generated by the TDC system designed in the University of Bristol Photonics Group. TDC had 1.87 ps LSB resolution, and therefore, the minimum digital delays and window sizes were 1.87 ps.

With this method, the smallest of window size with 0.12 ns able to capture all the coincidences generated by 512 KHz signals generated by an Analog Discovery signal generator. The results achieved with this implementation can be found in Appendix 8.3.3.2.

The biggest drawback of this implementation was the usage of a USB 2.0 interface which was hindering the real-time performance of the coincidence counting operation since time tags were needed to be downloaded from the FPGA first. However, it proved that the algorithm of forward looking tag difference was a suitable method of coincidence counting.

3.3 Backward Looking Tag Difference FPGA based Real-Time Coincidence Counter

3.3.1 Introduction

To overcome the USB throughput problem, the first attempt on migrating the software-based coincidence counter to the FPGA was made in this implementation. For this implementation, Opal Kelly XEM6310, which had Spartan 6 LX150 FPGA chip, was used. The main difference between this implementation and the final coincidence counter implementation was replacing the operation of the data structure used in the software approach with a RAM block in an FPGA. Therefore, the backward looking tag difference method was implemented with a BRAM block located inside the FPGA. For this project, Opal Kelly API used for implementing a software interface which could adjust channel delays, window sizes and program the FPGA. This design had

CHAPTER 3. LEGACY COINCIDENCE COUNTERS

two main parts which were TDC and the coincidence counter. These two parts will be discussed in this section. The system overview can be seen in Figure 3.2.

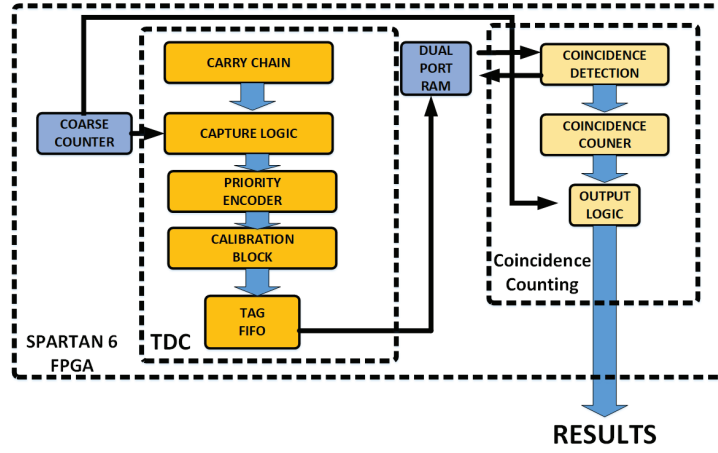


Figure 3.2: The block diagram of backward looking tag difference coincidence counter

3.3.2 Time-to-Digital Converter (TDC) Implementation

A tapped delay line TDC which was based on Xilinx CARRY4 primitives was used in this implementation. The implementation was featuring the followings :

- 256 bin Tapped Delay Line.
- 1.95 ps LSB.
- Average 39.8 ps resolution.
- 125 MHz Coarse Counter (8 ns period).
- 4 channel time tag generation.

In this implementation, a 256 bin delay line was utilised where the priority encoder was used to generate 8-bit code. The system also had a code density calibration for calibrating the TDC. However, it was software-based, and for each operation, the calibrated codes for the look-up table was loaded from the PC to FPGA. After the calibration, fine codes were scaled up to 12-bit code to apply a smoothing effect on the measurements, and however, the average resolution was 39.8 ps. The average resolution in this implementation referred to the average bin width across the delay line and precision was measured as part of the experiments. For this implementation, every other bin of the 4-bit CARRY4 block was utilised. This approach mainly aimed to achieve better linearity and reduce the need for a calibration. However, the improvement in the linearity resulted in a reduction of precision since a single delay size was

3.3. BACKWARD LOOKING TAG DIFFERENCE FPGA BASED REAL-TIME COINCIDENCE COUNTER

doubled. In this implementation, 32-bit coarse and 12-bit fine tags were used which were generated from parallel running 4 TDC blocks.

3.3.3 Coincidence Counter Implementation

The coincidence detection and counting were performed by iterating in a RAM block. For this operation, a dual-port RAM was utilised where two different memory pointers were used for reading and writing. The purpose of this RAM was storing the time tags in a time-sequential manner and looping through these time tags for counting the coincidences. The algorithm simply set the coincidence window at the time tags to read at the memory location and checks the time difference between the tag and the next available tag in the buffer. Once a tag difference was measured as more significant than the coincidence window, the window was shifted to the next tag. Based on the channel identifier of time tags the coincidence patterns were formed, and when, the tag difference was larger than the window size was detected the coincidence counter address in a RAM block was incremented by one. The integration time of this design was set 500 ms, and every 500 ms results were sent to the PC to be displayed.

Although this was a plausible solution to an integration of a TDC and a coincidence counter into the same FPGA fabric, this implementation had some serious drawbacks. These drawbacks were memory pointer collision issues and the buffer overflow. Memory pointer collision happens when the read and write pointers point the same memory location in the RAM block, which resulted in a loss of data for that memory address. This issue happened at least once for a complete iteration of the RAM block, and thus, some of the coincidences were missed because of this issue. The RAM overflow was caused when the window size was too large. Thus, until the coincidence counting for the window ends, some of the time tags will be overwritten in the RAM by the newer tags. Hence, this resulted in data loss, and as window size became larger, the supportable input rate dropped.

3.3.4 Summary

The first integration of coincidence counter with TDC in the same FPGA fabric was achieved in this implementation, and where, the backward looking tag difference method was used. This method was implemented by using a RAM block as a buffer to store the time tags, and the coincidences were searched inside this buffer.

The TDC implementation had 1.95 ps LSB resolution, but a 256 binned delay line was used, and it achieved an average of 39.8 ps resolution. With using this method, different coincidence windows sizes were tested, and the results can be seen in Appendix 8.3.3.3.

The problem observed in this method was a large distortion at the coincidence rates. This was likely to be caused by two different reasons; the first one was pointer collision

happens while traversing inside the tag buffer and the second one was dropping some of the tags due to the slow processing of the buffered tags. These results showed that this method was not ideal for a real-time coincidence counting, and thus, alternative methods were needed to be researched for implementing a coincidence counter without using a tag buffer.

3.4 FPGA based Forward Looking Tag Difference Coincidence Counter using Tag Serialisation

3.4.1 Introduction

After a problematic attempt on implementing a coincidence counter, finally, tag buffering was abandoned to improve the real-time operation. This method was inspired by the published work in [14], where the multi-stop LiDAR system was implemented using a multi-stop TDC. Therefore, the multi-stop time differencing method was used to implement a coincidence counting operation. In this implementation, the coincidence counter unit was fed by a FIFO storing the time differences between time tags from each channel. Eventually, this method was also abandoned for implementing a fully scalable and integrated real-time TDC based coincidence counter. In this section, the details of this implementation will be discussed — the results achieved from this method also available in Appendix 8.3.3.4.

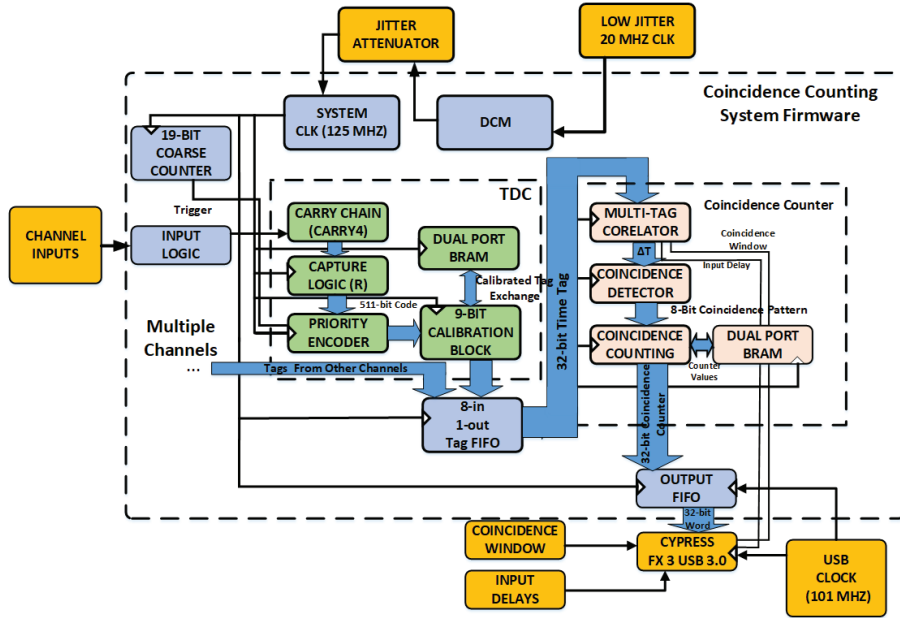


Figure 3.3: The system overview of tag serialising coincidence counting system

3.4. FPGA BASED FORWARD LOOKING TAG DIFFERENCE COINCIDENCE COUNTER USING TAG SERIALISATION

3.4.2 System Overview

As it was published in [7], the integration of the coincidence counting and TDC was implemented using a FIFO. This method implemented 8 TDC channels which were connected to coincidence counter through an 8x1 FIFO. This method implies an easier implementation since only one coincidence counter was required. However, 8 times more dead time was introduced during this process. TDC calibration and multi-tag correlator modules used in this system were almost the same as the ones used in the final design. The system possessed 8 channels of TDC modules with their correlators and the calibration blocks and a single coincidence detection and coincidence counting module. The system was implemented on the Opal Kelly XEM6310 board which had Spartan 6 LX150 FPGA and the breakout board designed in the University of Bristol which was also used for the final design. These details will be discussed later in this thesis. However, the implementation details of TDC and coincidence counter will be summarised in this section, and the old architecture can be seen in Figure 3.3.

3.4.3 Time-to-Digital Converter (TDC) Implementation

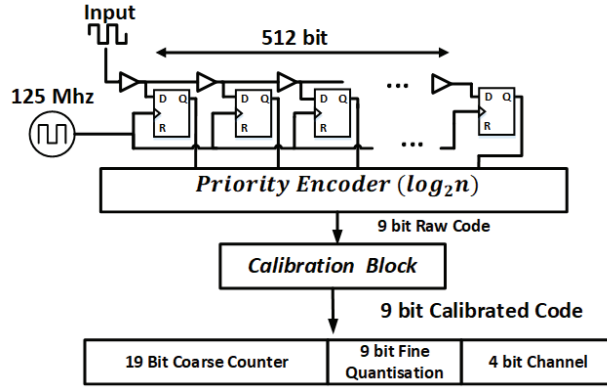


Figure 3.4: The tag generation for tag serialising TDC [7, 9]

A 512-bin carry chain was used to implement a fine TDC where the SSP was around 12.8 ps, and the LSB resolution was 15.6 ps. The system clock and the coarse counter was running at 125 MHz. The jitter inherited in DCM's Delayed Locked Loops [201] reduced by an external jitter attenuator to achieve high precision. Tags had 19-bit coarse counter, 9-bit fine tags and 4-bit channel identifier. Also, on-the-fly calibration, which was first introduced in [14], was used for this implementation, and on-the-fly calibration will be discussed in the next chapter. The tag generation scheme for the implementation can be seen in Figure 3.4.

After the tag generation, a multi-tag correlator module which, was used in the final design unmodified, applied multi-stop time differencing method to time tags. This

method measured the delta times between one START signal and many STOP signals. This was first described in [14], and it has been also used unchanged for the final design.

3.4.4 Coincidence Counter

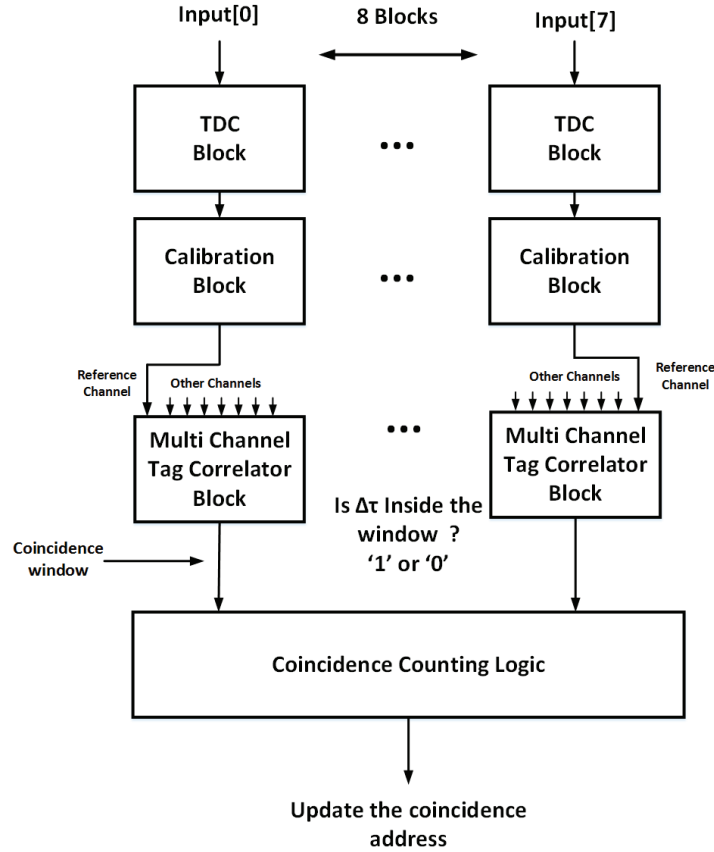


Figure 3.5: *Illustration of coincidence counting scheme [7]*

The coincidence counter took a single tag from 8-to-1 FIFO, and the coincidence detection scheme immediately compared the received delta times with the coincidence window (T_w) which was used for forming coincidence patterns and the addresses of the coincidence counting RAM. Unlike the final implementation of this module, it only used a single RAM block to achieve this. However, the coincidence detection and counting principles were implemented in the same way as the final proposed system's, and it will be discussed in the later chapters with more details. A diagram for this coincidence counting scheme can be seen in Figure 3.5.

Additionally, since coincidences were all counted together, duplicates were an issue in case of coincidence folds being larger than 2. The final implementation correctly

addressed this issue by implementing independent and parallel coincidence counters.

3.4.5 Summary

Among legacy coincidence counters, this implementation was very similar to the proposed coincidence counting system apart from the channel labelling scheme in the final design. This implementation used the forward looking tag difference method by utilising the tag correlator module, which was continuously finding the time differences between time tags to one channel. Then, these differences were used for comparing them with the coincidence window.

This method achieved 15.7 ps LSB resolution TDC with 12.8 ps SSP. The smallest window observed to be capturing every possible coincidence was 150 ps. These results can be seen in Appendix 8.3.3.4. This method connected the TDC with a coincidence counter using an 8-to-1 FIFO.

The main drawback of this design was the tag serialiser used in the form of a FIFO. The FIFO introduced a cycle delay for each additional channel in the TDC. Therefore, since the 125MHz clock used for 8 channel operation $8 \text{ ns} \times 8$, 64 ns dead-time would be introduced. This limited the single-channel count rate with 15 MCPS. Therefore, in order to improve the count-rate of the coincidence counter, a complete parallel multi-channel coincidence counting system was developed, which will be discussed in the further chapters of this thesis.

3.5 Conclusion

In this chapter, the properties of the previously designed coincidence counters were covered. Each design's limitations and the ideas behind them were discussed. Results achieved from each implementation can be found in Appendix 8.3.3.

To sum up, the first coincidence counter was implemented using both forward looking tag difference and rolling window implementations on software. For the time tagging an FPGA based TDC was used. This method's main problem was the USB-2.0 interface which was used for downloading the tags from a TDC board, and it was hindering the real-time performance.

The second coincidence counter managed to integrate a coincidence counter and TDC into the same FPGA fabric, which was implemented using the backward looking tag difference method. However, this method achieved distorted coincidence counting results which were likely to be caused by the pointer collision of write read operations and overflowing buffer due to slow data processing.

The third iteration was very similar to the final version of the coincidence counting system, and the forward looking tag difference method was implemented in an FPGA by using multi-tag correlators. This method removed the tag buffering for the coincidence

CHAPTER 3. LEGACY COINCIDENCE COUNTERS

counting and coincidences were detected as they were generated. However, this method was problematic due to the serialisation FIFO placed between the coincidence counter and TDC, which was introducing an extra dead-time and reducing the count-rate.

In the following two chapters, the implementation details of the final coincidence counting system will be discussed, and this chapter aimed to provide more clarity to the finalised system features by showing the stages of the development.

CHAPTER 4

Time-to-Digital Converter Implementation

4.1 Introduction

In this chapter, the details of the TDC implementation used to implement a multi-channel coincidence counting system will be discussed. In addition to the implementation details, the theory of some concepts will be expanded to provide a better understanding of the research. The implementation details include the TDC design, calibration module and trigger generation. Also, the concepts behind the code density testing method and averaging TDCs will be covered in more details in this chapter. It will start with the TDC design section, and each of the implementation detail will be covered.

4.2 Time-to-Digital Converter (TDC) Implementation

The proposed TDC implementation is based on a 512-staged delay line which was implemented by using Xilinx's carry chain primitives, CARRY4. Each CARRY4 block accommodates 4 bits adder, and thus, 512-bins utilises 128 CARRY4 block. The system clock runs at 125 MHz, and this delay line structure is utilised for subdividing the system clock into 512 equal bins. It is done by propagating the input trigger through the carry-in and out pins that are connecting the blocks. At the rising edge of the system clock by using D-type flip-flops, the values of the output pins are captured to record how far the trigger signal was propagated through the CARRY4 blocks.

CHAPTER 4. TIME-TO-DIGITAL CONVERTER IMPLEMENTATION

Essentially, this provides a digital representation of the trigger signal location to the clock edge. Hence, the dead-time of the carry chain is 8 ns. Unlike a typical delay line TDC, in the proposed scheme both clock edges are used for the fine tag generation. Thus, each trigger is represented by one code for the rising edge of the clock and one for the falling edge of the clock. Later, these codes are averaged for improving the linearity of the code [3].

The output of the carry chain is represented with a thermometer code, and where, the thermometer code consists of trailing '1's and '0's. Hence, when the N th bin is reached at the rising edge of the clock, the output will be N number of '1's and $512 - N$ number of '0's. In order to make use of this thermometer code, a priority encoder can be employed to convert it into a 9-bit digital code. Primarily, the priority encoder applies logarithm to the number of '1's in the code. This is done by checking the most significant high bit, and the encoder loops through the code to find the most significant bit. It should be noted since TDC operates with both clock edges two priority encoders, where one for the rising edge and one for the falling edge, are provided [3].

After the priority encoder step, tags are passed to the tag selector module where the detection of its clock region is identified. Once the clock region is detected, the averaging is applied to tag for improving the linearity. The averaged code results in 10-bits to provide a smoothing effect on the measurements. This 10-bit averaged code is passed to the calibration to apply the 10-bit code density calibration. After the calibration, the code can be used for the coincidence counting operation [3].

An illustration of tag generation process can be seen Figure 4.1. For how carry chains are used for a TDC implementation in an FPGA SLICE is shown in Figure .

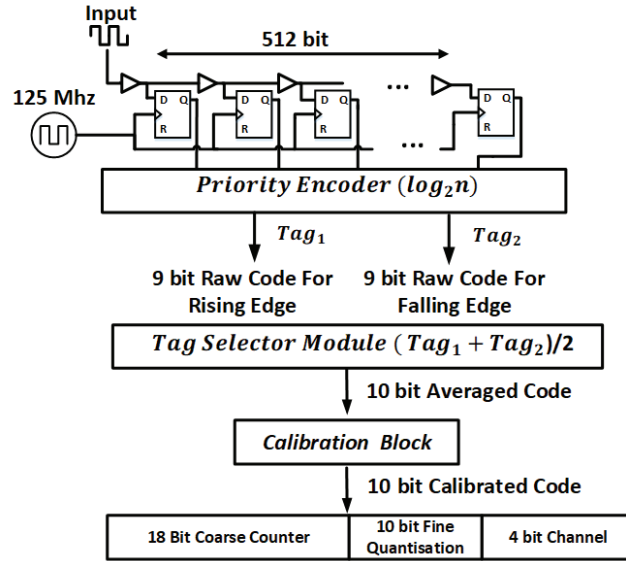


Figure 4.1: TDC tag generation process

4.3. TDC CALIBRATION AND LINEARITY IMPROVEMENTS

Since the 10-bit code is generated for 8 ns, the LSB of the code was set to the 7.7 ps (LSB resolution) The last recorded bin that could be reached in the delay line during the 8 ns clock period was recorded as 404 with the new TDC design, which corresponds to bin 810 for the 10-bit code. Therefore, the average resolution of the delay line is $8 \text{ ns}/808 = 9.9 \text{ ps}$.

The LSB resolution of the code can be expressed as below.

$$LSB = \frac{T_{Clock}}{2^N} \quad (4.1)$$

And it can be calculated as below.

$$LSB = \frac{8 \times 10^{-9} \text{ s}}{2^{10}} = 7.7 \text{ ps} \quad (4.2)$$

Once the TDC module generates a fine code for the trigger signal, linearisation and calibration steps are applied to the code consecutively. In the following sections, these steps will be discussed.

4.3 TDC calibration and Linearity Improvements

4.3.1 Double Data Rate Registration Linearisation

Averaging delay lines for improving the precision of the TDC has been implemented in different architectures. Delay lines are generally implemented by using a dedicated carry-chain or similar logic structures, and FPGA based methods, due non-linearity bin widths are varied across the delay line. However, since each delay line has different bin widths, averaging multiple delay lines effectively, reduces the sizes of wide bins. This can be implemented by either lengthening the delay line like in the double registration or using parallel multi-chains like in [159, 202, 155]. Regardless which implementation is used, the final code T_{final} can be formulated as below.

$$T_{final} = \frac{1}{N} \sum_{k=1}^N (T_{normal}(k) - T_{av}(k)) \quad (4.3)$$

In Equation 4.3, N is the number of delay lines, the T_{final} is the final time measurement, T_{normal} is the measured time at the k th delay line and T_{av} is the total average time until the k th delay line where, $1 \leq N$. Based on the chosen approach, this N can be stretched horizontally or vertically. When the delay lines stretched horizontally like in the double registration method, typically further dead-time is introduced due to multiple clock cycles is required for the delay line. On the other hand, when the delay line is stretched vertically, the space efficiency suffers. Illustration of averaging TDC using parallel delay lines can be seen in Figure 4.2.

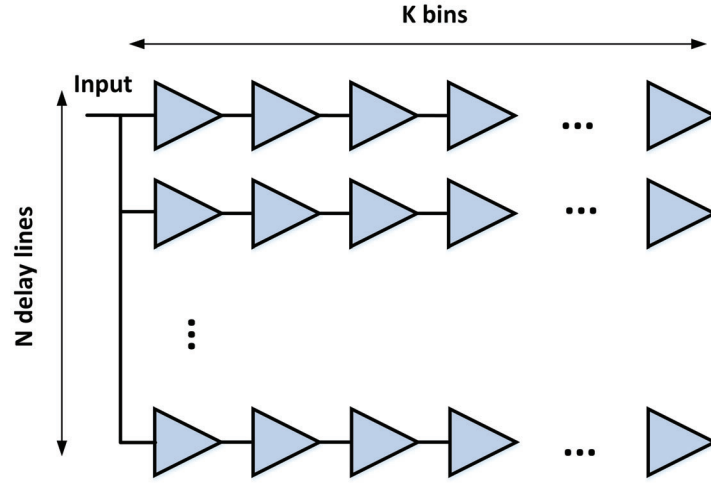


Figure 4.2: Multi-chain delay lines TDC

An example of an averaging TDC with serial delay lines can be seen in Figure 4.3, where the multiple clock cycle is used to quantise N number of delay lines.

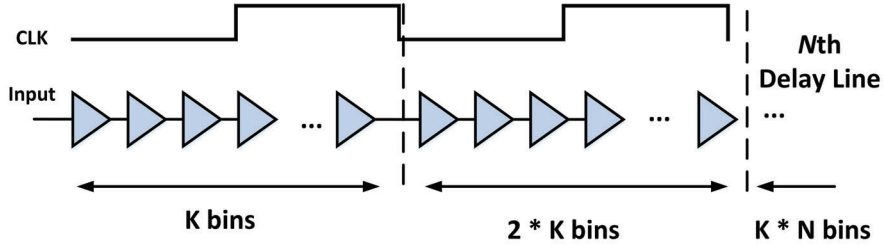


Figure 4.3: Multiple Times Registration TDC

Both averaging TDC approaches either sacrifices from the logic utilisation by using multiple delay lines or introducing extra dead-times by lengthening the delay line horizontally. In order to provide a similar improvement, in linearity by averaging methods, both clock edges can be used to quantise the same delay line. Since using both clock edges will extend the range of the measurement and averaging two value will effectively reduce the bin-widths around to their halves. This proposed method [3] is named as Dual Data Rate Registration TDC. This TDC method is used in time tagging process in the proposed system. The equation for the code generation can be seen in Figure 4.4,

$$T_{final} = \frac{T_{rising} + T_{falling}}{2} \quad (4.4)$$

where T_{final} is the linearised code and T_{rising} is the code generated by the rising edge of the clock and $T_{falling}$ is the code generated by the falling edge of the clock.

4.3. TDC CALIBRATION AND LINEARITY IMPROVEMENTS

The Dual Data Rate Registration is implemented by using the following steps in the proposed architecture,

1. The first step of the linearisation is capturing the state of the delay line both at the rising and the falling edge of the clock, which is done by two sets of D-type flip-flops.
2. Two captured 511-bit thermometer codes are passed to priority encoders.
3. For the both rising and falling edges of the clock, separate priority encoders are employed, and thus, two 9-bit fine codes are generated for two 511-bit thermometer codes.
4. After the codes are generated, they are passed to the tag selector module, where their clock regions are detected.
5. The tag selector module fixes the half clock period difference between two codes.
6. 1-bit is added to both codes, and two 10-bit codes are added up to find the average. This step essentially replaces the division operation by increasing the LSB resolution.
7. The 10-bit averaged code is passed to the calibration block for applying the code density calibration.

4.3.1.1 Tag Selector Module

Tag selecting process aims to detect the clock region of the trigger event. This module is necessary since both rising and falling edges of a code pair can be generated in different clock cycles. When the signal is detected in the rising edge region (when the clock is low, and the trigger is high), The both code pairs can be used at the same rising edge of the clock. Primarily, the rising edge generates a code much smaller than the one generated at the falling edge. When codes are generated for a trigger, they can be either in the clock low or clock high regions. In these regions, the distance between the trigger and the first utilised clock edge can be defined as ΔT . Therefore, the time between the trigger and the second utilised clock edge can be expressed as $\Delta T + T_{clk}/2$, where T_{clk} is the clock period. Illustration of this calculation can be seen in Figure 4.4 [3].

When the signal is in the falling edge region (clock high region), the generated code for the falling edge will be read by the system in the next rising edge since the system operates at the rising edge of the clock. This operation mainly introduces a cycle delay between codes generated in the falling and the rising edge of the clock. Therefore, whenever a trigger is detected in the falling edge region, the next rising edge of the clock is waited for being able to average the codes captured in both clock

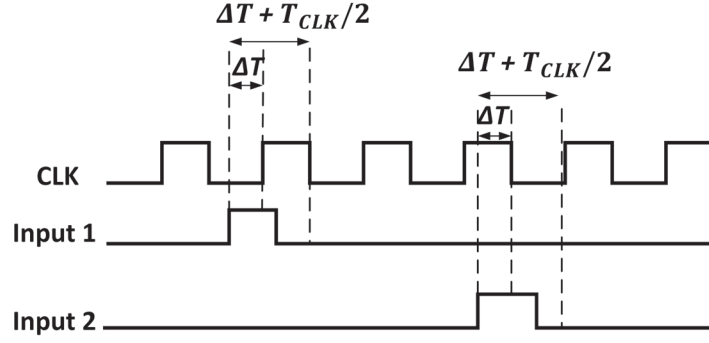


Figure 4.4: Example of dual clock edge quantisation for inputs

regions. However, when the trigger is detected in the rising edge region, the code for both rising and falling edges of the clock will be generated in the same clock cycle. Also, due to the half of a period difference between clock regions, it is needed to be taken into account for the final tag. Therefore, when the trigger is at the falling edge region, 256 is needed to be subtracted from the code, and when it is in the rising edge, region 256 is needed to be added.

The averaged code for the trigger before the rising edge can be calculated as below.

$$T_{Final} = T_{rising} + T_{falling} + 256 \quad (4.5)$$

Average code for the trigger before the falling edge can be calculated as below.

$$T_{Final} = T_{rising} + T_{falling} - 256 \quad (4.6)$$

It should be noted that for the equation above, T_{rising} and $T_{falling}$ both have additional 1-bit for this operation. Thus, T_{final} has 10-bits.

Another function of the tag selector is averaging the codes after the adjustments for the clock region is taken account. The averaging in digital implementation is simply by adding another bit of zero at the end and summing the two numbers. Thus, an addition of two 9-bit codes results in 10-bit averaged code, and the LSB becomes 7.7 ps. Although, the scaling improves the LSB resolution, it has no impact on SSP, and it is limited with a smoothing effect. After the code is averaged, the process is continued with the calibration.

4.3.2 Self Calibrating TDC

The use of a delay-line structure for fine time measurements leads to issues with non-linearities between delay elements. Non-linearities are mainly caused by the routing of the design inside the FPGA fabric, power and temperature changes. This leads to a non-linear sub-division of the system clock [14].

4.3. TDC CALIBRATION AND LINEARITY IMPROVEMENTS

In the Xilinx Spartan 6 FPGA architecture, each FPGA slice accommodates 4-bit carry elements named as CARRY4. Hence, for subdividing the clock period into 512-bins, 128 FPGA slices are required. Figure 4.5 demonstrates how the FPGA fabric connects FPGA slices using interconnects [14].

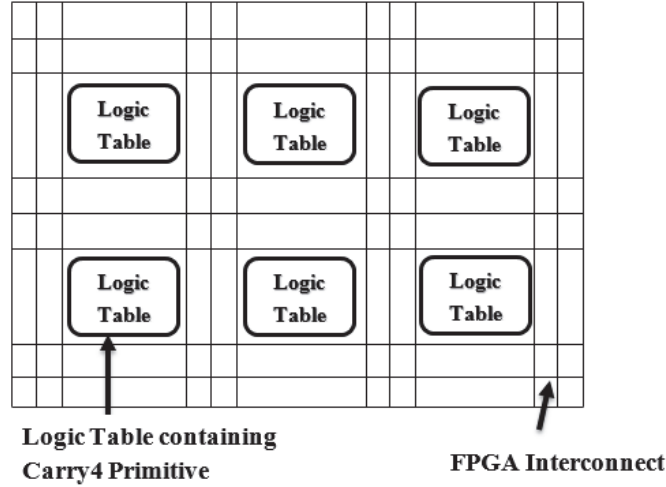


Figure 4.5: High-level overview of logical design layout within FPGA fabric [14]

The non-linearity caused by the interconnects between FPGA slices. CARRY4 blocks typically require FPGA interconnects to link the outputs of the adjacent delay-lines within the FPGA fabric. Thus, non-linear propagation delay spread was observed in the delay line, as it was shown in Figure 4.6 [14].

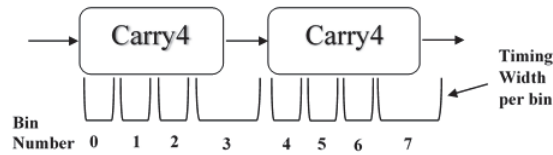


Figure 4.6: Non-linearity within delay-line structure [14]

4.3.2.1 Code Density Testing Calibration

In statistics, the probability density function expresses the probability distribution for random variables. Random variables obtained by a measurement, and determined by a range rather than discrete values are defined as continuous random variables. The probability density function gives the likelihood of random variable being in a range of values at a given point. For instance, for continuous random variable X , the function

CHAPTER 4. TIME-TO-DIGITAL CONVERTER IMPLEMENTATION

$f_X(x)$, where the density of values of X at the point x can be expressed as below [203].

$$f_X(x) = \lim_{\tau \rightarrow 0^+} \frac{(P(x) < X \leq x + \tau)}{\tau} \quad (4.7)$$

The probability density function of continuous random variable X at the point x can be obtained as the derivative of the CDF $F_X(x)$. The probability function of X can be defined as the equation below [203].

$$F_X(x) = \frac{dF_X(x)}{dx} \quad (4.8)$$

CDF can also be acquired from the integral of the PDF. Therefore, the CDF of the continuous random variable X , $F_x(x)$ can be formulated as below.

$$F_X(x) = \int_{-\infty}^x f_x(v) dv \quad (4.9)$$

Therefore, for the $P(x < X \leq x + \tau)$ CDF can be derived as below [203].

$$F_X(x) - F_X(x + \tau) = \int_x^{x+\tau} f_x(v) dv \quad (4.10)$$

As the time is a continuous variable, the probability of a pulse being in a certain bin during a clock period can be used to characterise the delay line bin widths. Ideally, every bin should have an equal chance of getting hit and split the clock period into equal bins. Thus, it should present an uniform distribution, although it is not reflected on the bin widths due to non-linearities. When bin widths are tested with a number of known asynchronous pulses, the probability of each bin getting hit will be proportional with the bin widths. Thus, larger bins will have higher counts, and the bin counts versus the total number of counts will reflect the PDF of a single bin. Later, applying the CDF on each bin's PDF will provide the transfer function of the delay line.

When the bins are tested with inputs asynchronous to the system clock, the accumulation of the hits of each bin can reveal the bin's probability density among the other bins. Therefore, the bin width of i th bin B can be expressed as in equation below.

$$B(i) = \frac{T_{clk} * Y(i)}{K} \quad (4.11)$$

Where T_{clk} is the clock period, Y is counts for a specific bin and K is the total number of counts. By applying the CDF to the equation given above, the following transfer function H for a delay line can be formed.

$$H(i) = \sum_{i=0}^K B(i) \quad (4.12)$$

The concept of code density calibration effectively uses the properties of PDF and CDF on the bin widths to mitigate the effects of the non-linearity [14, 155].

4.3. TDC CALIBRATION AND LINEARITY IMPROVEMENTS

In the proposed system, the calibration was used at the system start-up. Ideally, a perfectly linear delay line would provide, equal distribution of bin widths (white Gaussian noise). However, as a result of non-linearities, the larger bins have a higher probability of getting hits while, the shorter bins records fewer hits as it can be seen in Figure 4.7 [14]. Also, in Figure 4.7 around bin 350 extraordinarily large non-linearity has been observed. These large delays were potentially caused by a non-ideal mapping of the design, and subsequently resulted in large inter-slice propagation delays between CARRY4 blocks which were located very far away from other delay elements.

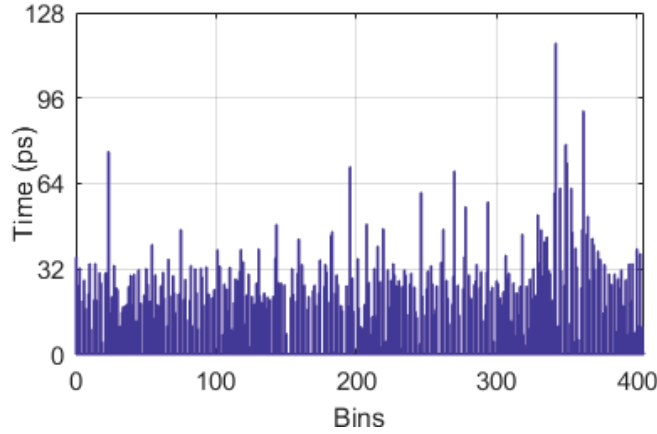


Figure 4.7: *Bin widths across delay-line*

To reduce this effect, a calibration scheme which tries the delay line with a known number inputs which are asynchronous with the system clock is utilised. With the calibration, bin widths are statistically analysed for their distribution amongst other bins. The system used an FPGA Block Random Access Memory (BRAM) to record the distribution of bin counts across the delay line per TDC channel, and then, cumulatively adds the distributed counts across the delay line, and records the cumulated statistics within an FPGA BRAM, as shown below in Figure 4.8 [14, 155].

Each bin's fine time representation can be generated from the comparison of the counts cumulated for the bin with the total distributed counts across the delay line. This provides the sub-division of the system clock by each individual delay line bin. The following equation expresses this representation [14].

$$Bin\ Width = T_{clk} \times \frac{N_{cumulative}}{N_{Total}} \quad (4.13)$$

An FPGA BRAM is utilised as a look-up table to acquire the calibrated fine-tags for each bin, and therefore, upon registering a trigger event in the system, the bin number is input as a memory address for the calibration BRAM and the calibrated fine-tag is obtained. The calibration step is repeated at each system start-up since

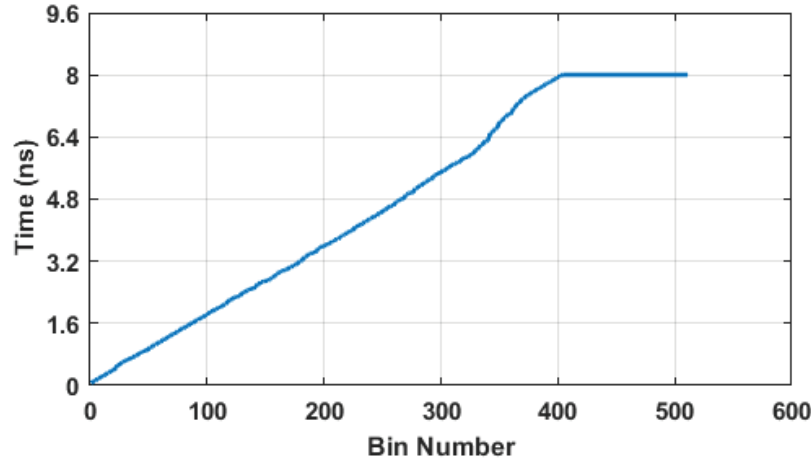


Figure 4.8: *Cumulative distribution of counts across delay-line structure*

each run can have different mapping on the FPGA fabric and different environmental temperatures [14, 155].

The algorithm is implemented in the following steps ;

1. Initially, the system is held at a soft reset mode, which prevents tags from leaving the TDC block during the calibration.
2. Then, with trigger signal, non-calibrated tags are passed to the calibration block.
3. The non-calibrated codes passed from the tag selector are used as the index of a RAM block where RAM block has 1023 depth with 12-bit width.
4. Upon receipt of non-calibrated tags by the calibration block, the 12-bit counter located inside the RAM block's address is accessed and incremented by one.
5. After incrementing the counter, total bin counter which is a 16-bit number, is incremented by one.
6. From steps 2 to 4 are repeated until the 16-bit total bin counter reaches to 2^{16} , hence the total of 65536 triggers is required for characterising the delay line.
7. After total bin number is reached, CDF is applied to the contents of the RAM.
8. In the CDF step, the content of N th RAM address is summed by the content of the $N - 1$ and written back to address N .
9. The previous step is continued until $N = 1023$.
10. After CDF is finished, the RAM is ready to be used as a look-up table, and thus, the soft reset is lifted.

11. During the operation, when a trigger is registered by the system, the calibration block is used as a look-up table to provide the calibrated codes.
12. Calibrated codes are passed to the multi-tag correlator module for used in coincidence counting operation.

4.3.3 Trigger Generation

A typical TDC tapped delay line would be using D-type flip-flops to capture the state of the delay line, and this implies the state of the delay line typically can be only captured with the single edge the system clock. Thus, the ideal dead-time of a TDC should be the system's clock period (T_{clk}), and however, it is not always the case since it strictly depends on how the trigger signal is generated.

Although this is a conceptually simple method, in implementation, it is rather complicated. The main issue with the trigger generation is input triggers are very likely to be wider than the clock period, which can be detected as multiple triggers. Hence, it is necessary to capture the rising edge of the trigger signal. The simplest way to achieve this is simply by delaying the signal a clock period and checking whether the delayed signal and the recent signal is different from each other. However, this method requires two clock cycles since an additional delay is needed to be introduced. Alternatively, D-type flip-flop, which is in-sync with the trigger signal, can be utilised. However, the flip-flop will need at least another clock period to refresh the output; otherwise, the output will be observed as one wide signal. Also, the first bin of the delay line cannot be used as the trigger for the cases where the trigger signal is wider than the clock period. An example of how the system typically recognises consecutive triggers can be seen in Figure 4.9.

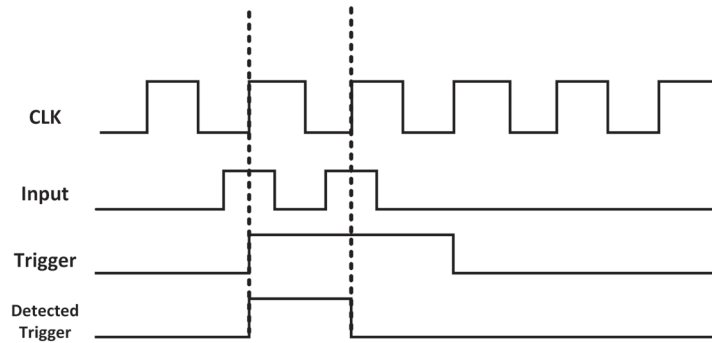


Figure 4.9: Waveforms of the trigger problem when the trigger is synchronised with clock edge

A plausible way to generate a trigger signal, which can be high for only a clock period, is employing a coarse counter. In this process, the counter is needed to be in-sync with the trigger signal, and each time the rising edge of the signal is detected,

CHAPTER 4. TIME-TO-DIGITAL CONVERTER IMPLEMENTATION

this counter is incremented by one. By checking the change in the counter's value, the trigger signal can be generated. Hence, if the counter's value is different from the previous clock cycle, a trigger is generated. Also, using a grey code for a counter implementation likely to avoid metastability issues can occur during the operation.

In the TDC design, whenever a trigger happens, the counter's value is changed, and by checking this change, a trigger can be generated. Also, using a grey code counter can make it possible to process more than a tag in a delay line since by merely checking how many bits have changed, the number of triggers can be counted. However, further changes would be required in the priority encoder for this method.

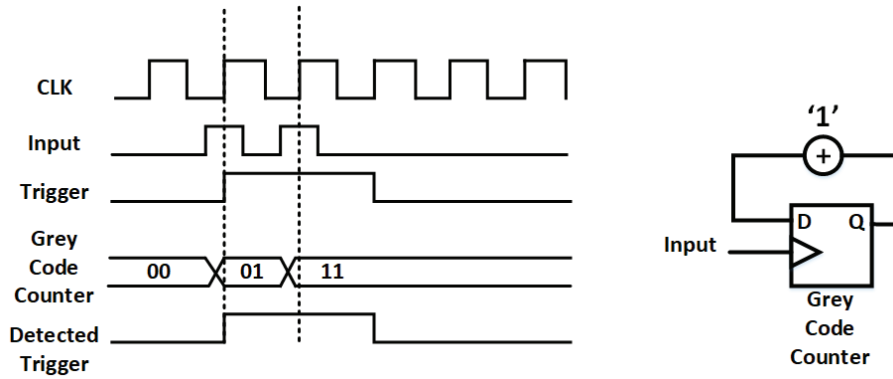


Figure 4.10: Waveforms when the trigger is generated by the counter

Although, the coincidence counting's count rate is limited by the RAM operation, a good trigger generation scheme is necessary for achieving the optimum count-rate from the TDC. Thus, in the future, if the coincidence counter's RAM operation is improved, this trigger generation method could become useful.

4.4 Summary

To sum up, in this chapter the TDC implementation details used in the coincidence counting system was discussed. These details include how to carry chains were used for implementing a 512-bin TDC, the Dual Data Rate Registration Method was developed, and the self-calibrating TDC was implemented by using the code density calibration. Also, the theory behind the Dual Data Rate TDC and The Code Density Testing method was discussed in more details in this section. The implemented TDC provided 7.7 ps LSB resolution, and on-the-fly calibration was implemented in the hardware. To improve the linearity of the TDC, the Dual Data Rate Registration Method was developed, which essentially captures the delay line with both clock edges and applies an averaging method similar to multi-chain and double registration TDCs. Additionally, a trigger generation logic for improving the count-rate was discussed in this section.

In the further chapters, the results achieved for this TDC will be discussed. The thesis will continue with the details of the coincidence counting design.

CHAPTER 5

A Precise High Count-Rate Multi-Channel Coincidence Counting System

5.1 Introduction

In this chapter, the implementation details of the multi-channel coincidence counting system will be discussed. The implementation details include the multi-tag correlator, coincidence detector and coincidence counter modules and the algorithms implemented in these modules to achieve their operations. Also, the hardware details used in this research, the software used for the USB 3.0 interface and the post-processing will be discussed. The implementation details will start by explaining the system overview, and the details of other parts will be discussed later in this chapter.

5.2 System Overview

The coincidence counting system consists of three essential parts which are the TDC, coincidence counter and output FIFO. The principles of the TDC was discussed in the previous chapter. Mostly, the coincidence counting uses the time tags generated by the TDC to detect the coincident events. The coincidence counting principle uses the channel labelling coincidence counter for parallel counting and the forward looking tag difference for forming the coincidence patterns. The coincidence counter consists of 3

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

different modules which are multi-tag correlator, coincidence detector and coincidence counter.

The objectives of each module inside the coincidence block can be listed as below :

- **Multi-Tag Correlator** : Generating the time difference between the time tags (ΔT).
- **Coincidence Detector** : Using ΔT 's for forming the coincidence patterns.
- **Coincidence Counter** : Using the coincidence patterns as addresses of the counter RAM and counting the coincidences.

The results obtained from coincidence counting block are sent to the Personal Computer (PC) at the end of each integration time by the output FIFO. This FIFO is controlled by the Cypress FX3 chip, which is also responsible for providing channel delays and coincidence window to this module from PC. The overview and the interaction of modules in the coincidence counter can be seen in Figure 5.1.

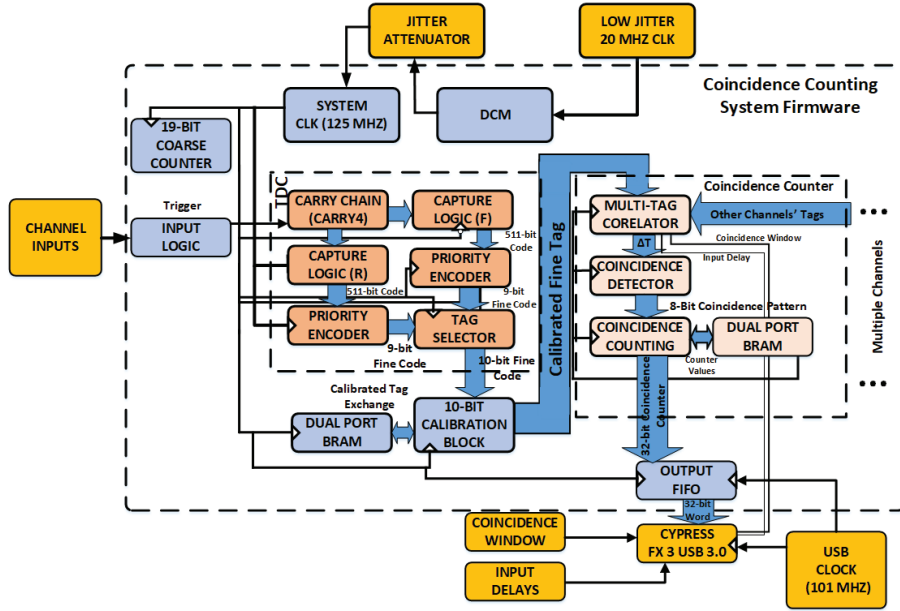


Figure 5.1: Components of the coincidence counting system

5.3 Coincidence Counter Implementation

The coincidence counting operation follows a time tag generation. The coincidence counting consists of four different modules, and each module has different roles in the operation. These modules are the tag correlation, coincidence detection and coincidence

5.3. COINCIDENCE COUNTER IMPLEMENTATION

counting modules. These modules are scaled with the number of channels present in the implementation, and they all run in parallel. In this system, the number of channels are 8, hence each of these modules has 8 parallel counterparts. The integration time for the coincidence counting was set 60 ms and controlled by the counter value, which was incremented by one every 8 ns . This counter value could be changed for different integration times, but for a fast data display on the PC, this number should be kept relatively small.

The key features of the coincidence counting system can be listed as below :

- 8 parallel channel operation of coincidence counting and time tagging.
- Adjustable input delays and coincidence windows between the range of 7.7 ps - 2.06 ms.
- Forward looking tag difference method coincidence detection.
- Multi-channel labelling method for avoiding duplicate counting.
- No buffering and serialisation between TDC and coincidence counter.

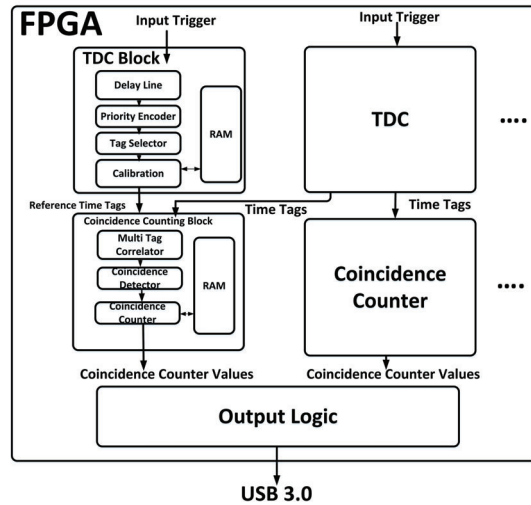


Figure 5.2: The overview of coincidence counting system

A summary of the FPGA logic utilisation of the system was recorded as follows: 11,636 Slices (50%), Register 24,926 (13%), Look-up-table (LUT) 3,329 (36%) and 85 I/O (100%). The system used in Spartan 6 slx150fpg484-2n can be summarised from 4 aspects which are the Slice Logic Utilisation, Slice Logic Distribution, IO Utilisation and Specific Feature Utilisation. The designed system utilised 25753 out of 184304 Slice Registers (13%) and 36329 out of 92152 (39%) of Slice LUTs. For the Slice Logic

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

Distribution, a total of 47875 flip-flop pairs were utilised. In terms of special features, RAM/FIFO utilisation of the system was 22 out of 268 (8%), the number of global clock buffers used was 8 out of 16 (50%), and the PLL utilisation was 1 out of 6 (16%). This logic utilisation report points out; the system still has room to add further complexity or additional channels. However, it should be noted that having available space in the FPGA fabric does not guarantee meeting with timing constraints when further complexity is added to the design. Therefore, lesser space available in the FPGA fabric hardens the meeting with timing constraints.

5.3.1 Multi-Tag Correlator

Multi-tag correlation module mainly applies multi-stop time differencing method to the time tags from its channel perspective, which was presented in Chapter 2 as the Multi-STOP LiDAR. Thus, the tags generated from its channels are used as the START signal and all other tags coming from other channels as the STOPS. This results in the contentious generation of delta times between the START and STOPS. These delta times are passed to the coincidence detection module. Also, any digital delay introduced by the software is added to the channel in this stage [14].

When the calibration is completed, the multi-tag correlator becomes idle and starts waiting for tags to arrive. It processes any subsequent time tags received from TDC modules. The predefined reference channel is set, and it remains the same during the operation. Upon the arrival of a new tag, tag's channel identifier is checked to determine its source channel. Based on this identifier, the next operation is decided. Either the START tag is updated (if the tag is received from the reference channel), or delta-time is calculated for the distance between the START and STOP [14].

Since each reference signal is constant, the delta time calculation can be expressed as below:

$$\Delta_n = (T_{STOP_n} - T_{START}) \quad (5.1)$$

The explanatory diagram for the delta time calculations of the tag correlator can be seen in Figure 5.3

The operation of the multi-tag correlator follows the following steps,

1. Initially, it stays in the reset mode until the calibration ends.
2. Once the calibration is done, it starts waiting for tags in the idle.
3. The channel identifiers of the received tags are checked to determine whether a START or STOP received.
4. If the START is received then the tag is stored for future calculation.

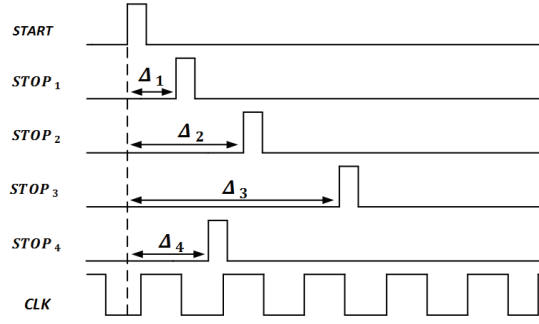


Figure 5.3: *Illustration of delta time measurement for multiple stops by the tag correlator [14]*

5. If the STOP is received the ΔT is calculated by subtracting it from the tag stored for the START.
6. After ΔT is calculated, it is passed to the coincidence detector, and steps 4 and 5 are repeated until the end of integration.

The coincidence detectors follow the multi-tag correlator for the coincidence detection from the tag differences.

5.3.2 Independent Coincidence Detectors

The coincidence detection logic follows the tag correlation. Coincidence detection is a process where the delta times are checked for being within the coincidence windows. The bits of the coincidence patterns are set by checking each tag with 8 independent coincidence detector. Thus, each coincidence between channels can be checked from the channel's perspective without being limited by the data serialisation. Each generated coincidence pattern is used as the address of the counter, which corresponds to the index of the coincidence counting block.

The coincidence algorithm forms coincidence patterns based on generated tags falling within a coincidence window between two trigger signals of the reference channels. Thus, the trigger on the reference channel acts as the reset for the coincidence pattern. The coincidence pattern is formed based on comparing the tags with the coincidence window, and when, a tag is detected within the window, the bit corresponding to its channel is denoted by a logical one in the coincidence pattern, otherwise, it is kept in logical zero. The coincidence detector aims to find the largest coincidence fold possible and leaves the sub-fold finding to the post-processing step. The equation for the coincidence pattern formation can be seen in Equation 5.2.

$$V(n) = \begin{cases} 0 & \text{if } \Delta T_n \geq T_w \\ 1 & \text{if } \Delta T_n < T_w \end{cases} \quad (5.2)$$

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

The coincidence detection uses the forward looking tag difference method to form coincidence patterns. This detection process follows the steps below.

1. Once the delta time is received by the coincidence detector the channel identifier is checked.
2. If the channel identifier is equal to the reference channel, the coincidence pattern is reset to "00000000".
3. If the channel identifier is not equal to the reference channel, then the difference is compared with the coincidence window ($\Delta T < T_w$).
4. When $\Delta T < T_w$, the bit corresponding to the channel identifier of the ΔT is set to '1' else to '0'.
5. This pattern forming continues until the previous module multi-tag correlator informs that a START signal has been received, and then, the pattern is sent to the coincidence counting.
6. operation continues with step 2 unless the integration time is over.

It should be noted that until a tag from the reference channel is received this operation continues, and thus, the largest coincidence fold between two START signals is formed as a product of this operation. The operation continues with the coincidence counting.

5.3.3 Independent Coincidence Counting Blocks

Each independent coincidence counting block has its own RAM block which holds the counter values for each channel. Each channel has a 511 deep and 32 wide BRAM, which accommodates counters for all possible coincidence patterns. When a coincidence pattern is detected, it is used as the index of the RAM block, and the counter located in the address is read and incremented by one.

Having independent coincidence counting blocks makes the coincidence counting independent from other channels, and the data serialisation is no longer required, unlike [7]. This essentially makes the data-rate of the coincidence counting operation independent from the channel numbers. Thus, the coincidence counting scheme becomes scalable. The coincidence counting operation continues for an integration time which is set to 60 ms. When the integration times are finished, the results are sent out, and the contents of the BRAM is cleared. A block diagram for the coincidence counting block can be seen in Figure 5.4, where M represents the RAM block's memory and the index of the M represent the coincidence address.

5.3. COINCIDENCE COUNTER IMPLEMENTATION

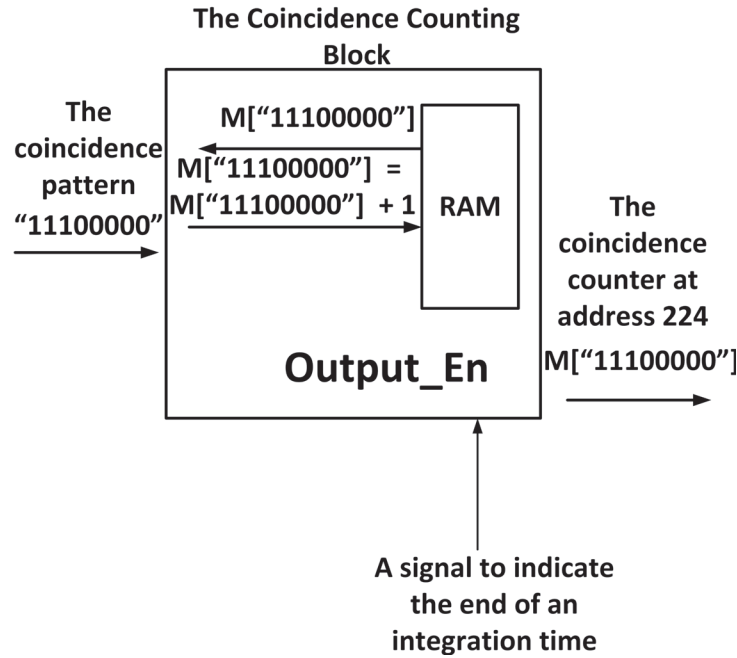


Figure 5.4: Illustration of coincidence counting block

The algorithm of the coincidence counting block can be split into two phases. The first phase can be coincidence counting operation, and the second phases are the output of the content of the RAM at the end of an integration period.

The first phase of the algorithm has the following steps :

The Coincidence Counting :

1. With the detection of the START signal, the coincidence counter passes the coincidence pattern to the coincidence counter module.
2. The coincidence pattern is used as the address of the counter located at the memory location of the RAM.
3. The next clock cycle the content of the RAM is read and incremented by '1'.
4. After incrementing the counter, The RAM is switched to the writing mode, and the incremented counter is written back to the coincidence address.
5. If the integration time (60ms) is not over, the operation continues and goes back to step 1. Of the integration, time is over phase 2 proceeds.

The second phase initiates with the end of the integration time and follows the steps below :

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

Outputting the Coincidences :

1. Once the output enable signal is received, the output mode is enabled.
2. First, the RAM is switched to the read enable mode; then the address is set to the "00000001" which is the first element in the RAM.
3. 8 coincidence blocks concurrently send 32-bit counter values at the addresses of their RAMs.
4. By using a FIFO 256-bit to 32-bit, the content of the RAM at the memory location is sent out.
5. After all the channels' RAM contents at the address is sent out to the PC, indices of the coincidence counter blocks' RAMs are incremented by one, and data in the next RAM locations are loaded to the FIFO.
6. The process of loading the RAM contents to the FIFO for every channel continues until address 255 is reached.
7. Once all the RAM contents are sent out, results are displayed on the screen and the soft-reset is sent to the FPGA from the software.
8. The soft-reset prevents the TDC from the system reset, and thus, the calibration look-up table is not changed, but the coincidence RAM and other signals in the operation are reset for the next operation.
9. Then, the operation continues with phase 1.

It should be noted at least 3-clock cycles are needed for updating the coincidence values located inside the RAM because of the RAM writing operation. This is inevitable since RAM needs to switch between Reading and write modes. This means it needs to wait until the content of the RAM is available to be incremented and written back to the RAM. These 3 clock cycles in a Spartan 6 architecture are where 1 cycle for accessing the content of the RAM, 1 for switching the mode and incrementing the content and 1 cycle for switching back to the write mode and updating the RAM address. Also, the data readout is from a the same RAM, hence there is readout dead-time. The readout dead-time can be estimated as 20.4μ since USB clock runs around 100 MHz (10 ns) and 8 RAM blocks with 256 address are read during the operation. From readout dead-time system throughput can be estimated. Throughput can be defined as the effective transmission rate [204], and thus, can be calculated in coincidence counting application by multiplying the theoretical count rate with the

5.3. COINCIDENCE COUNTER IMPLEMENTATION

ratio of effective integration time versus theoretical integration time. The calculation of throughput can be seen as below.

$$Throughput = \frac{Effective\ Integration\ Time}{Theoretical\ Integration\ Time} \times Theoretical\ Count\ Rate \quad (5.3)$$

The ideal integration time is 60 ms and effective integration period is 59.9796 ms(60 ms - 20 μ s) and theoretical max channel count rate is 41.67 MCPS. Therefore, a single channel data throughput is :

$$Throughput = 0.9996 \times 41.67\ MCPS \quad (5.4)$$

$$Throughput = 41.65\ MCPS \quad (5.5)$$

Once the coincidences are sent to the PC, the post-processing step, where duplicate and missed counts are calculated, is proceeded. This post-processing calculation will be discussed in the next subsection.

5.3.4 Sub-fold Coincidence Identification Algorithm and Channel Labelling

Running independent coincidence counting operations in parallel raises a problem of detecting the same coincidence pattern multiple times from different channels or not detecting sub-fold coincidence patterns since the coincidence detection are designed to find the greatest coincidence counting pattern. As a result, some coincidence folds get counted more than they occur. To fix this problem, a labelling scheme has been implemented for each channel and an algorithm to discard the patterns, which were counted extra or add the ignored sub-fold patterns in the post-processing step.

These situations can be seen in Figure 5.5. Where channel A detects 3-fold coincidences from its perspective, and there are two 2-fold patterns which are "110" and "011" and they need to be detected from channel A's perspective while channel B detect a two-fold between B and C from its perspective. However, without labelling the coincidence pattern, "011" will be detected twice from two different channels.

The post-processing implements 3 operations which are :

- **Updating the sub-fold counts from the super-fold's counts :** Since the FPGA algorithm prioritises the largest coincidence pattern, sub-folds located inside these patterns are ignored and needed to be calculated in the post-processing step.
- **Summing the counts from all the channels :** Since each channel has coincidences counted from different channels perspective, for displaying results, the coincidences are needed to be summed up.

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

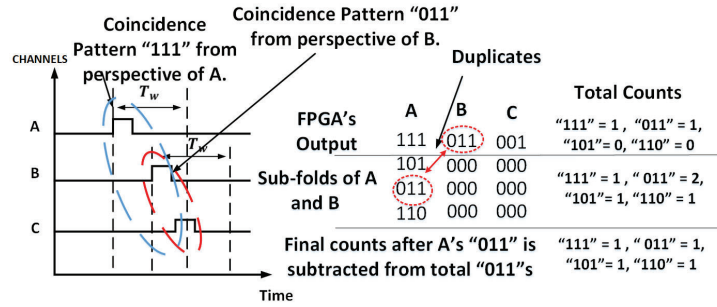


Figure 5.5: The demonstration of coincidence fold detection from multiple channels

- **Discarding duplicates** : Since the first step guarantees that all the sub-folds are calculated from each channels perspective, the totalling operation results in some duplicates when the same coincidences are detected from different channels. Thus, these duplicates are needed to be discarded.

These three post-processing functions are implemented by the following algorithm steps :

1. Initially, at the end of an integration time coincidence values are received by the PC through the USB 3.0 interface where the data is decoded.
2. Once the data is usable by the software, the coincidences are stored in different arrays based on their detection channel since each channel detected separately by the system.
3. When all the 255 counter values for each channel is received, the algorithm loops through patterns for finding the sub-folds.
4. The sub-fold finding is essential finds the smaller coincidence patterns located inside, the larger coincidence patterns, and update their counts with their super-set patterns.
5. After the sub-fold address is updated, all channels' coincidence values are summed for the total counts.
6. The summation results in some duplicates, and for this purpose, the larger coincidence pattern's count is compared with its sub-folds' counts.
7. If the sub-fold's count is greater than the super-fold's, the super-fold's count value is subtracted from its sub-fold's for discarding the duplicate.
8. This operation is done in a loop. Once all the addresses are covered, results are displayed on the terminal screen.

5.4. COINCIDENCE COUNTER HARDWARE

9. After the display, the soft reset is passed to the FPGA, and the next integration time starts.

This post-processing step was essentially designed for finding the total counts of the coincidence system without missing counts or counting them multiple times. Thus, for displaying coincidences from each channel's perspective, only the sub-fold finding process is necessary, and the rest of the operation can be skipped. Also, since the integration time is over, it has no impact on the count-rate of the coincidence system.

5.4 Coincidence Counter Hardware

The system was implemented on the Opal Kelly XEM6310 board which was accommodating Xilinx 45 nm Spartan 6 LX150 FPGA. Each Opal Kelly XEM6310 provides 100 MHz clock, Cypress FX3 USB 3.0 interface, 128 Million Bits (MiB) Dual Data Rate (DDR) 2 Synchronous Dynamic Random Access Memory (SDRAM), 16 MiB FPGA Flash and two SamTec Expansion Connector. A block diagram for the Opal Kelly XEM6310 can be seen diagram Figure 5.6 and features in Table 5.1.

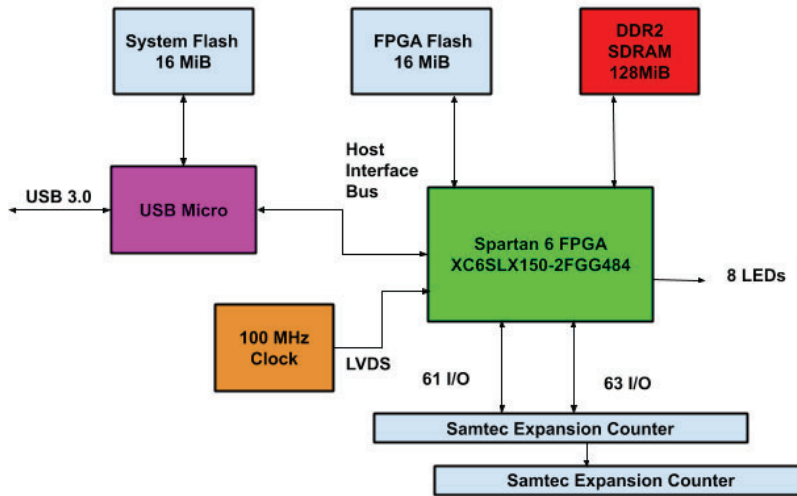


Figure 5.6: *The Opal Kelly XEM6310 System Components [2]*

In addition, in the University of Bristol Photonics Group, a breakout board was designed for the Opal Kelly XEM6310. Opal Kelly XEM6310 mounts the breakout

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

Features	XEM6310-LX150
FPGA	XC6SLX150-2FGG484C
SLICE	23.038
D Flip-Flops	184,304
Distributed RAM	1,355 Kib
Block RAM	4,824 Kib
DSP Slices	180
Clock Management Tiles	6

Table 5.1: *Spartan 6 LX150 Features [2]*

board through the Samtec Connectors and uses components accommodated on the board. The components on the breakout board includes jitter attenuators, 25 MHz low jitter clock, 8 Sub-Miniature A (SMA), Input/Output (I/O) connectors, Digital-to-Analogue Converter (DAC), input comparators. The ICs located on the breakout board are listed below :

Breakout Board Hardware

- IQD IQXT-210 25 Mhz Low Jitter Clock.
- Texas Instruments LM340MP-5 Voltage Regulators.
- IDT ICS874001I-02 Jitter Attenuators.
- Analog Circuits AD5671R DAC.
- Texas Instruments LMH7220 High Speed Comparator with Low-voltage Differential Signaling (LVDS) Output.
- SAMTEC SAMTEC80 connectors.

The system clock used in this system is originated from 25 MHz low jitter differential clock from the oscillator located on the breakout board for this purpose IQXT-210 oscillator, which has a period jitter of 2.4 ps [205], used. The low jitter clock is input through LVDS I/O located on a Spartan 6 FPGA and Xilinx digital Digital Clock Manager (DCM) is used to multiply the clock to 125 MHz clock. Once the clock is multiplied, it is output for the jitter attenuators. For this purpose, the clock is output through the Xilinx Output Dual Data Rate (ODDR) core to the attenuator IC located on the breakout board. The jitter attenuators located on the board was IDT ICS874001I-02 which provided 3 ps RMS jitter [206]. After the jitter attenuation is applied to the clock, it is input back to FPGA through LVDS port and used as the system clock.

The board also accommodates the DAC IC, which controls 8 sets of a comparator. These components primarily apply required safety and filtering for the voltage of the inputs. DAC sets the threshold voltages is required at the inputs and comparators

5.5. COINCIDENCE COUNTER SOFTWARE INTERFACE

provides 3.3 V when the input voltage is above a certain threshold. FPGA programmes DAC and the software interface uses the Opal Kelly Front Panel Application Program Interface (API). The threshold values passed to the FPGA through the USB 3.0, and they are written to DAC by using Xilinx I2C cores. Then DAC keeps the thresholds constant until it is set otherwise. The board also provides 8 channels of LVDS output, and the DAC controls the voltage on these ports. However, this feature was never used since it was not needed for coincidence measurements.

The picture of the Opal Kelly board with the breakout board can be seen in Figure 5.7,

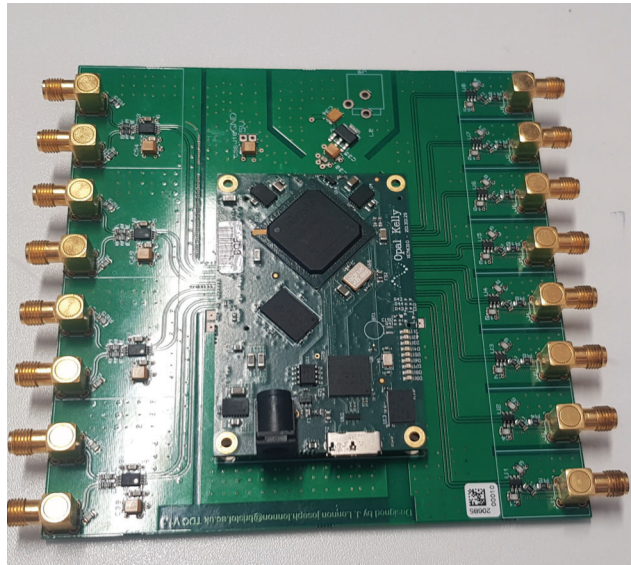


Figure 5.7: *The Opal Kelly XEM6310 Board with the breakout board*

5.5 Coincidence Counter Software Interface

The data processed within the FPGA is needed to be transferred to the PC to collect the results and display them on the monitor screen. Also, some required post-processing is applied to counts on the software interface part of the system. Software interface is essential, a terminal application was written in C++ to interface the USB 3.0. The USB interface communicates with FPGA through Cypress FX3 USB 3.0 chip. The USB carries 32-bit wide data with 101 MHz clock which corresponds to around 400 Mbit/s data rate over USB 3.0.

The software interface operation includes sending the voltage threshold values for the comparators, programming the FPGA, setting digital delays, setting the window size, controlling the hard and soft resets, decoding USB packages, writing the

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

recorded coincidences to files from each channel's perspective and displaying the total coincidence counts on the terminal screen. The features of the software interface can be listed as below.

The Software Features :

- Decoding the received data from the USB 3.0 interface.
- Storing the received coincidence data and labelling them based on their received channels.
- Programming the FPGA with the bit file.
- Sending individual digital delays for each channel.
- Controlling the soft and the hard resets for the system.
- Displaying the total coincidences on the terminal screen, and writing each channel's coincidences to files.
- Controlling the voltage thresholds for the comparators by programming the DAC.
- Applying the post-processing algorithm to correct the coincidence counts for the final results.

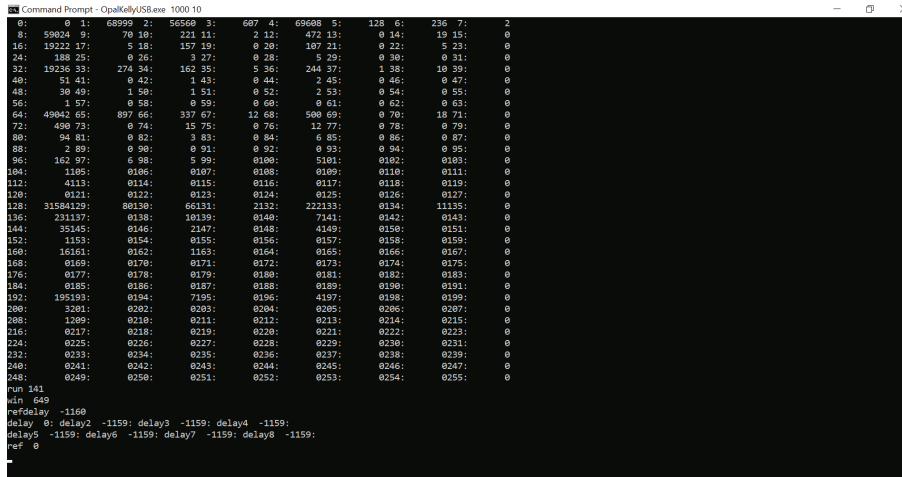
The software interface was written by using the Opal Kelly Front Panel API [207]. Front panel API specifies 3 types of connection types between the FPGA and PC sides. These are Pipes, wires and triggers . Pipes are used for a continuous large sequence of data to transfer through a FIFO to the FPGA synchronously. The word size is 32-bit long in the pipe type. The pipe types are used for sending the coincidence data to the PC. The second type is the wire in and outs [207]. These asynchronous connections are used for setting the window size, delay values and signals such as resets. The third type is triggers, which are asynchronous bit triggers, and used for triggering events in the FPGA. For instance, a trigger signal was used to trigger the DAC to start its operation [207].

One of the advantages of using TDCs for coincidence measurement is the possibility of tuning the channel delays digitally for each channel. The USB interface sends 32-bit long delay information to FPGA side, and inside the FPGA this value is added to tags. Also similarly, the window size is passed through the software to FPGA by this interface. A 32-bit wide window size is written by using input wires of the Opal Kelly API [207]. In addition, wires are used for the setting the soft and the hard resets. The soft resets is a type of reset that blocks FPGA from sending any data and holds coincidence counting operation while keeping the TDC running. The soft reset is

required for keeping the TDC calibrated, but resetting the coincidence data for the next integration time. Also, during the TDC calibration, the system is kept at the soft reset to prevent coincidence counter from sending any data to the PC. On the other hand reset, resets every aspect of the system and clears all the registers used.

The coincidence data is sent at the end of each integration time over the USB 3.0. The output FIFO, which runs with 101 MHz USB clock is used. Each data received consists of 24-bit count data and 8-bit pattern identification. This code is decoded by the software, and the data is calculated. Data recorded by channels are sent separately. Thus, the occurrence of the same pattern is counted and labelled based on the counter. For instance, the first coincidence data for the first RAM address is labelled as the label 1's address 1, while when the same data is received for the 8th time, it will be recorded as the label 8's address 1. This operation continues for all 8 labels and 255 RAM addresses. Once the data transfer is over, the soft reset is applied to the system, and coincidence operation is initiated again.

These received tags are displayed on the terminal screen as total coincidences, and the coincidence information for each coincidence address is based on their detection perspective. Every label's coincidence data is written into separate files. These files can be accessed upon the end of each integration time. The look of a terminal screen can be seen in Figure 5.8.



```

Command Prompt - OpalkellyUSB.exe 1000 10
0: 0 1: 68999 2: 56568 3: 607 4: 69688 5: 128 6: 236 7: 2
8: 58024 9: 70 10: 221 11: 2 12: 472 13: 0 14: 19 15: 0
16: 19222 17: 5 18: 157 19: 0 20: 107 21: 0 22: 5 23: 0
24: 188 25: 0 26: 3 27: 0 28: 5 29: 0 30: 0 31: 0
32: 19236 33: 274 34: 162 35: 5 36: 244 37: 1 38: 10 39: 0
40: 51 41: 0 42: 1 43: 0 44: 2 45: 0 46: 0 47: 0
48: 30 49: 1 50: 1 51: 0 52: 2 53: 0 54: 0 55: 0
56: 1 57: 0 58: 0 59: 0 60: 0 61: 0 62: 0 63: 0
64: 48042 65: 897 66: 337 67: 12 68: 580 69: 0 70: 18 71: 0
72: 480 73: 0 74: 15 75: 0 76: 12 77: 0 78: 0 79: 0
80: 94 81: 0 82: 3 83: 0 84: 6 85: 0 86: 0 87: 0
88: 2 89: 0 90: 0 91: 0 92: 0 93: 0 94: 0 95: 0
96: 162 97: 0 98: 0 99: 0 100: 0 101: 0 102: 0 103: 0
104: 1185: 0186: 0187: 0188: 0189: 0118: 0111: 0
112: 4113: 0114: 0115: 0116: 0117: 0118: 0119: 0
118: 0121: 0122: 0123: 0124: 0125: 0126: 0127: 0
118: 31584129: 80138: 60131: 2132: 222133: 0134: 11135: 0
116: 231137: 0138: 10139: 0140: 7141: 0142: 0143: 0
144: 35145: 0146: 2147: 0148: 4149: 0150: 0151: 0
152: 1153: 0154: 0155: 0156: 0157: 0158: 0159: 0
160: 16101: 0162: 1163: 0164: 0165: 0166: 0167: 0
168: 0169: 0170: 0171: 0172: 0173: 0174: 0175: 0
176: 0177: 0178: 0179: 0180: 0181: 0182: 0183: 0
184: 0185: 0186: 0187: 0188: 0189: 0190: 0191: 0
192: 195193: 0194: 7195: 0196: 4197: 0198: 0199: 0
200: 3201: 0202: 0203: 0204: 0205: 0206: 0207: 0
208: 1209: 0210: 0211: 0212: 0213: 0214: 0215: 0
216: 0217: 0218: 0219: 0220: 0221: 0222: 0223: 0
224: 0225: 0226: 0227: 0228: 0229: 0230: 0231: 0
232: 0233: 0234: 0235: 0236: 0237: 0238: 0239: 0
240: 0241: 0242: 0243: 0244: 0245: 0246: 0247: 0
248: 0249: 0250: 0251: 0252: 0253: 0254: 0255: 0
run 161
bin 640
msdelay -1160
delay 0: delay2 -1159: delay3 -1159: delay4 -1159:
delay5 -1159: delay6 -1159: delay7 -1159: delay8 -1159:
per 0

```

Figure 5.8: Terminal output for the coincidence counter

5.6 Summary

To sum up this chapter, the implementation details and algorithms used to implement a coincidence counter was discussed in this chapter. The coincidence counting opera-

CHAPTER 5. A PRECISE HIGH COUNT-RATE MULTI-CHANNEL COINCIDENCE COUNTING SYSTEM

tion was implemented with modules including the multi-tag correlator, coincidence detector and coincidence counter. The multi-tag correlator module was responsible for calculating the delta time, while the coincidence detector checks coincidences with the calculated delta time, and the coincidence counter counts them by using a RAM block.

Also, the software interface and hardware details used in this research was discussed. For this research, Opal Kelly XEM6310 FPGA board, which was mounted on a breakout board designed in the University of Bristol, was used. The breakout board provided a low jitter clock, jitter attenuators and 8 channel I/O. The software interface for this project utilised the USB 3.0 by using the Opal Kelly API, and the post-processing was implemented to discard the duplicate counts and add the miscounts.

The thesis will continue with the results achieved from the TDC implementation in the next chapter.

CHAPTER 6

Functionality, Linearity and Precision of the Time-to-Digital Converter

6.1 Introduction

In this chapter, the results achieved from various tests conducted on TDC modules located inside the coincidence counting system were presented. These tests mainly investigated the TDC's precision, linearity and the multi-stop functionality. These tests were necessary to show the reliability of the coincidence counting operation. Also, the improvements after the linearisation was presented in this chapter.

These tests were timing jitter measurement, DNL/INL tests and Multi-STOP testing. The timing jitter test is used to measure the jitter spread around the timing measurement, which can be used to obtain the SSP of the instrument. DNL/INL measurements are necessarily the characterisation of the bin widths of the delay line and its transfer function. Therefore, these measurements are essential to scrutinise the improvement in the linearity after the Double Data Rate Registration applied to the TDC. Multi-STOP tests aim to show the system capable of capturing multiple STOP signals from the perspective of a single START. This functionality is vital for coincidence counting since it is the essence of the real-time multi-channel operation.

This chapter starts with results obtained from the timing jitter measurements and continues with the other tests mentioned above.

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

6.2 Timing Jitter Measurements

The test known as a timing jitter measurement is concluded to benchmark the precision of the TDC. In this test, two identical signals are used to feed the TDC channels, and the routing delay between the channels is characterised. Essentially, the small routing delay act as a fixed delay. By characterising the jitter, the standard deviation affecting measurements, the SSP and the Full-Width-Maximum can be calculated.

From the results, SSP and FWHM of the measurements can be calculated. In this test, two TDC channels were fed by the same input signals, which was asynchronous to the system clock and where there was a fixed delay between channels. Then the differences between the time tags of the channels were characterised. For this measurement in order to achieve the best possible benchmark, the low jitter clock located on the Opal Kelly Board was used. Since this clock was asynchronous to the system clock, it was usable for this test. Hence, the effects of calibration and linearisation concepts was successfully scrutinised.

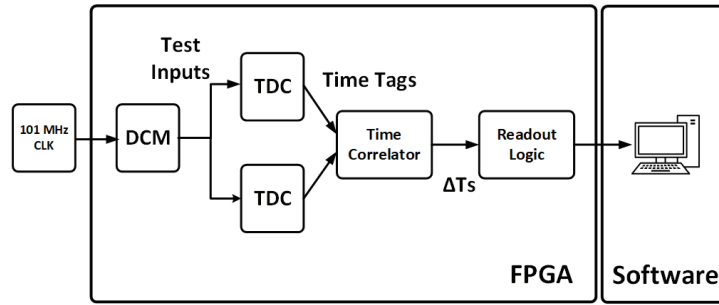


Figure 6.1: *The timing jitter test experimental setup*

The timing jitter measurements were obtained for three different iterations of the TDC. These were the non-calibrated TDC, calibrated TDC and the linearised and calibrated TDC. The time difference between the tags was measured using the multi-tag correlator module inside the FPGA, which was designed to measure the ΔT s between events. Also, measuring the time differences inside the FPGA eliminates the chance of dropping a tag by the FIFO during the data transfer and minimises errors could affect the tags.

The test setup features can be listed as below.

Timing Jitter Test Setup :

- The 100 Mhz Clock located on the Opal Kelly XEM6310 board used as the asynchronous source for the Xilinx DCM to generate 3.12 MHz signals.
- 3.12 MHz fed to different TDC channels with 50000 tags.

6.2. TIMING JITTER MEASUREMENTS

- The Tag correlator module utilised channel 1 as a START and channel 2 as the STOP signal and the time differences were calculated.
- The time differences were passed to PC and Matlab command *histdist()* was used plot and fit the best fit-line.

In Figure 6.1 the experimental setup for the timing jitter test can be seen. Since internally generated high speed signals are used for the tests, the input circuitry time walk was ignored. Thus, these tests provide TDC scheme's precision without the influence of the TDC hardware.

This test was conducted with 3 different TDC implementations. The first test was conducted using the non-calibrated TDC. Where the fine raw tags captured by the delay line and subtracted from the next coarse counter value without attempting any calibration or linearisation, the results can be seen in Figure 6.2. The SSP was measured as 22.06 ps, FWHM of the Gaussian Plot was 73.4 ps, and the standard deviation was 31.2 ps.

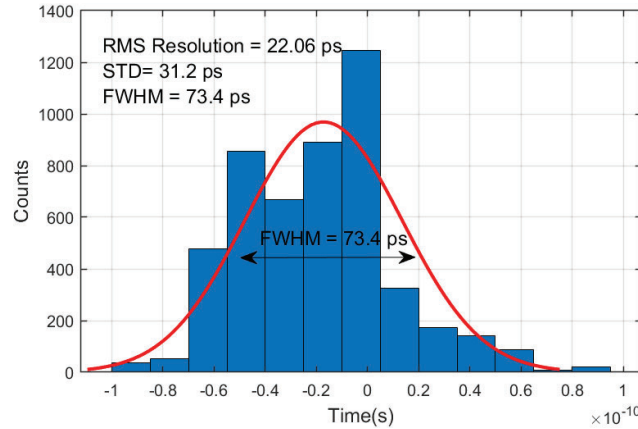


Figure 6.2: Timing jitter measurement with non-calibrated TDC

After calibration was attempted, the timing jitter was measured as it was in 6.3. The SSP was improved to 12.1 ps from 22.06 ps, the FWHM dropped to 40.2 ps from 73.4 ps, and the standard deviation was improved from 31.2 to 12.6. Around 1.9 times improvement in every precision parameter.

After the Dual Data Rate linearisation applied to the calibrated delay line, the results obtained from this test can be seen in Figure 6.4. The SSP was measured as 8.9 ps; standard deviation was 12.1 ps, FWHM was 29.6 ps. The improvement achieved from this measurement was observed as a root mean square of the non-linearised values. These results match with the results obtained from the previous research conducted on multi-chain averaging TDCs [159]; hence, by using both clock edges the same effect was observed at the results.

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

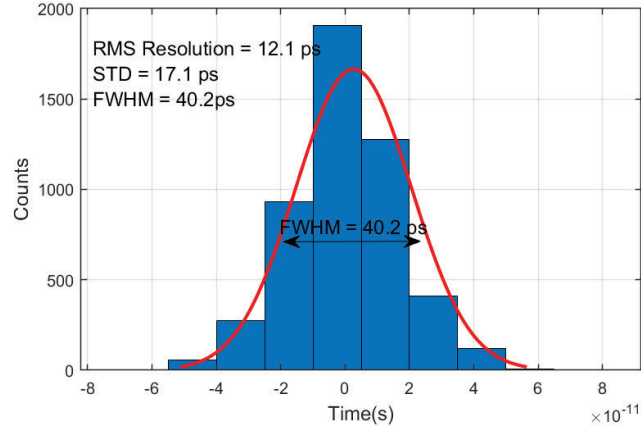


Figure 6.3: Timing jitter measurement with the calibrated of the TDC

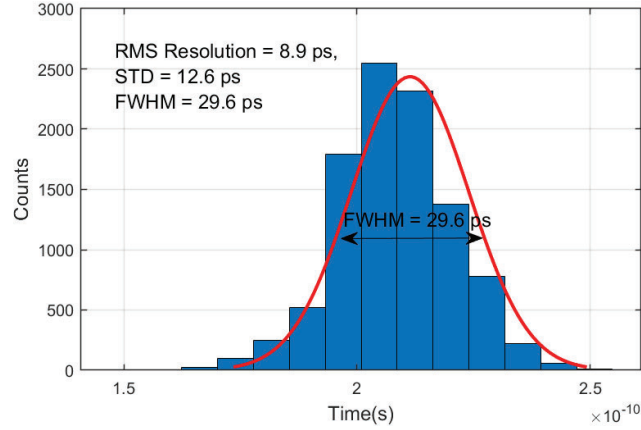


Figure 6.4: Timing jitter measurement with the linearised and calibrated TDC

6.3 Time-to-Digital Converter (TDC) Linearity

The TDC's linearity as it was discussed before reveals the information between the input and the output code. Ideally, this is expected to be linear, and thus, every input should have one unique output code but the non-linearities caused by various factors such as routing, temperature and power fluctuation, this is not possible in practice. In this section, the changes in the linearity as the calibration and the linearisation method were applied will be discussed. For this test B & K Precision BK4054B signal generator has been used the characterise delay lines, the input signal was set to 1 MHz. However, the source and the signal frequency was not that important in this test, since as long as the signal was asynchronous from the system clock, every bin's width can be characterised. The test setup for this experiment can be seen below.

6.3. TIME-TO-DIGITAL CONVERTER (TDC) LINEARITY

- B & K Precision BK4054B signal generator which was set to 1 MHz.
- A single isolated TDC block Firmware, and tags are sent directly to the PC through FIFO.
- 50000 tags were generated for each test.
- A software interface retrieved the tags from Opal Kelly XEM6310 board through USB 3.0 block.
- Bit-masking was applied to separate the coarse counter and fine tag from each other.
- Histogramming applied to the results with Hist() function, then the bins analysed for the parameters after the histogram was formed.

For every linearity measurement in this section, the setup given above was used. In the following sections, the bin widths, DNL, INL and transfer functions of the TDC will be discussed.

6.3.1 Bin Characterisation of the TDC

The bin widths of the delay line define the shortest measurable time by the TDC, and so, the resolution is directly related to it. Physically, these bin widths are limited by the propagation delays affecting each delay element, and they are unchangeable. However, as it was discussed before, these bin widths can be mathematically calibrated or linearised by applying some techniques. To understand the effect of these methods, first, the bin widths before the calibration and linearisation should be investigated. Since total of 50000 counts and 8 ns clock period were used, each count represent 0.16 ps ($8 \text{ ns} / 50000$). Therefore, by multiplying counts recorded for each bin with 0.16 ps, actual bin width can be calculated.

As it can be seen from Figure:6.5, the final reachable bin by 8 ns clock period was bin 405. The total number of empty bins were 207 out of 511, and non-zero bins were 304 out of 511. The mean bin width was calculated as 19.6 ps. up to the last bin. The ultra-wide bins are noticeable across the delay line. Around bin 350, bin widths are significant larger and two bins pass 600 counts which indicate 96 ps delays.

After the statistical code density calibration was applied to the delay line, the histogram was achieved as it was in 6.6. The mean bin width was calculated as 15.5 ps, which was equal to the $8 \text{ ns}/511$. Also, the number of empty bins was 218 out of 511, and non-empty bins were 293 out of 511. An improvement in linearity was observed as the large bins occurring after the bin 300 sprat across the delay line and the largest bin size dropped to around 72 ps (450 counts).

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

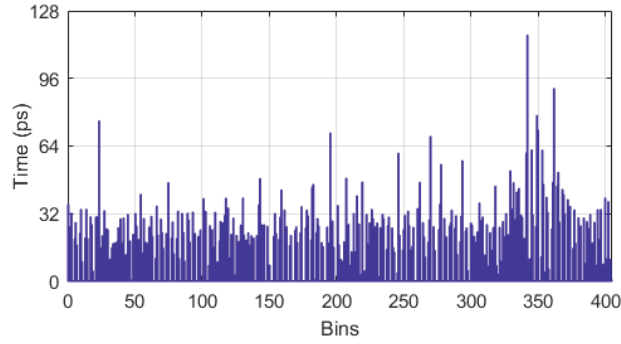


Figure 6.5: *non-calibrated bin counts of the delay line*

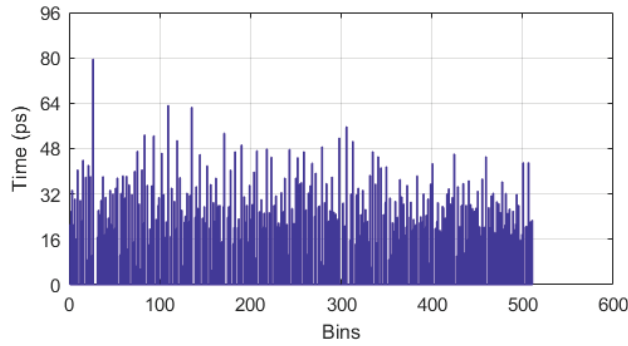


Figure 6.6: *Calibrated bin counts of the delay line*

After the linearisation, a significant reduction in the number of empty bins was observed where the total number of empty bins measured as 106 out of 511 and the non-empty bin number was 405 out of 511. In Figure 6.7, the impact of the linearisation can be seen. This was expected since averaging two different codes generates more variation in the codes. Also, the largest recorded bin was dropped to around 60 ps (380 counts).

However, the final code used was scaled up to 10-bit for applying for smoothing. When the same experiment was conducted for 10-bit codes, the number of empty bins observed as 283 and non-empty bins observed as 785 among a total of 1023. As it was expected these bin widths effectively split into approximately half. The division wasn't perfectly halving the bin widths since bin widths were not perfectly equal-sized. The new bin widths after the scaling can be seen in Figure 6.8. The bin widths are generally observed to be shorter than the 9-bit code and the largest bin width was recorded around 42 ps (350 counts).

After the bin widths were obtained DNL and INL of these histograms were calculated and they will be discussed in following sections.

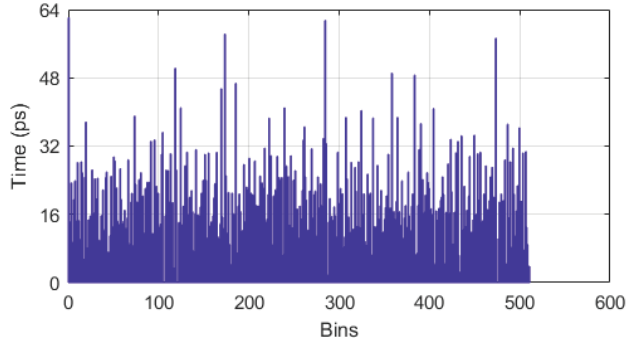


Figure 6.7: *Calibrated and linearised bin counts of the delay line*

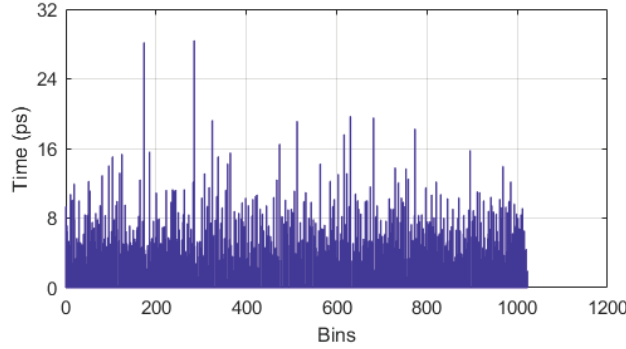


Figure 6.8: *Calibrated, linearised and scaled up bin counts of the delay line with 10-bit code*

6.3.2 Differential Non-Linearity (DNL) of the TDC

Ideally, every bin in the delay line should have an equal number of hits due to the equal-sized delay elements. However, this is never achieved because of the non-linearity issues caused by many factors mentioned before. The non-linearity appears as very large bins or missed bins. DNL error is a parameter which defines how much each bin deviates from the ideal step (1.249 LSB). The DNL improvement is also reflected on empty bin numbers in the code since averaging methods effectively splits bins into two by averaging them and increase the bin variation [3].

Also, it is necessary to have a clock speed which cannot reach the end of the delay line to make sure that it is long enough to subdivide the entire clock period. Therefore, the last reachable bin for the delay line before the calibration with the 125 MHz clock was approximately 405. This will appear as a high DNL error since a significant number of bins are never reached in the delay line, and even in the best-case scenario, there will be at least around 106 empty bins depending on where the last bin is.

Before the calibration, the largest DNL error appeared to be around 3.9 LSB as it

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

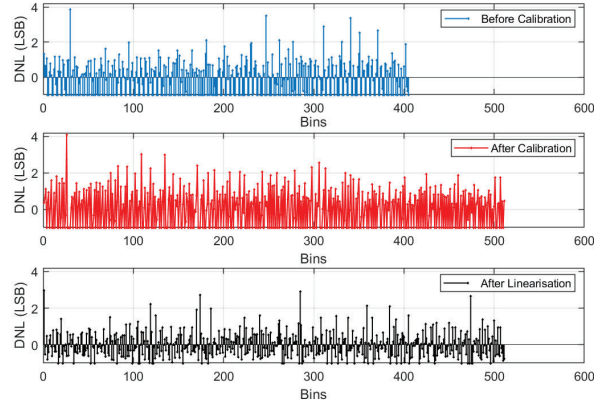


Figure 6.9: *The differential non-linearity before and after linearisation*

can be seen in Figure 6.9a. However, after the calibration the largest DNL error was observed as slightly larger with 4.1 LSB than the non-calibrated TDC and generally the DNL error was increased across the delay line since calibration process stretched the last bin of the code from 405 to 511. This stretching also resulted in an increase of number of empty bins.

After the linearisation, an improvement in DNL was observed as it can be seen from Figure 6.9. The maximum DNL error was dropped from 4.1 LSB to 2.9 LSB, and smaller DNL error was observed across the delay line.

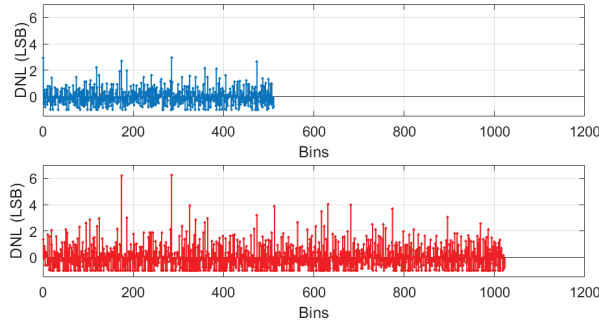


Figure 6.10: *The differential non-linearity before and after scaling*

When the scaling was applied, the new DNL error was observed as in Figure 6.10. The scaling typically doubles the DNL errors across 1023 bins since the bin widths were halved and the largest DNL error was observed around 6.1 LSB. Then the characterisation of the delay line continues with INL.

6.3. TIME-TO-DIGITAL CONVERTER (TDC) LINEARITY

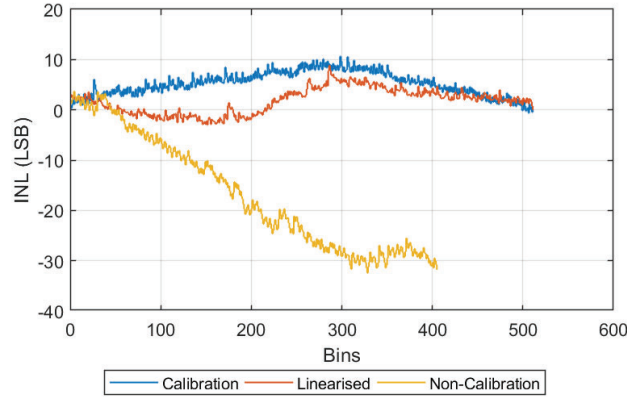


Figure 6.11: *The integral non-linearity before and after linearisation*

6.3.3 Integral Non-Linearity of the TDC

INL is also a measure of linearity in converter instruments. INL reveals the cumulative effect of the DNL across the delay line. Therefore, each INL value gives the total LSB error affecting at the bins and how far the transfer function is deviated from the ideal transfer function. The comparison of INLs for non-calibrated, calibrated, and linearised code can be seen in Figure 6.11.

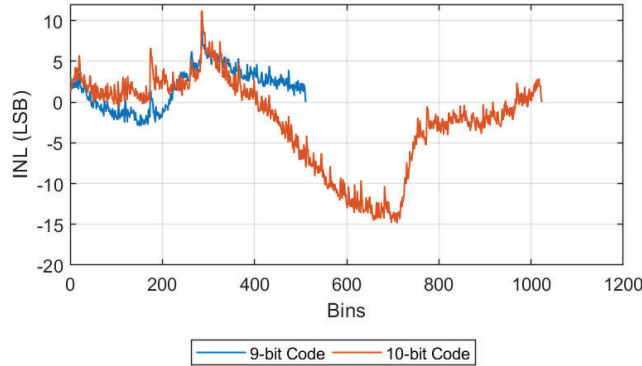


Figure 6.12: *The integral non-linearity before and after scaling*

As it can be seen in Figure 6.11, INL of non-calibrated delay line was observed much larger than the other two approaches. However, since the code density testing was calibrated the delay line by taking account of DNL, the transfer function was adjusted to have correct time steps for each code. Thus, calibration reduces deviation of each code steps from the ideal. When the linearisation was applied, effectively the large bin sizes are reduced by averaging bins in the different regions of the delay line,

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

and this resulted in a reduction in the DNL error. Since INL is a CDF of the DNL, after the calibration of the linearised code, an INL was observed as much closer to the zero than the previous approaches. The maximum INL error observed before the calibration as -31.8 LSB. After the code density calibration, the maximum INL observed as 10.64 LSB, and with the calibration and linearisation combined, the maximum INL error was measured as 8.8 LSB. Thus, around 1.8 LSB max INL error improvement was observed.

When the scaling was applied to the bin widths, a similar INL graph with a double in width and the height was expected since it was mostly had a smoothing effect. In Figure 6.12, the comparison of 9 and 10-bit codes can be seen, The result was similar to what was expected, and the max INL error was around -15 LSB which was around twice of the 9-bit code's max INL error but in the opposite direction. -15 LSB observed around bin 650-700 which corresponds to bin 325-350 in 9-bit code where all the INL graphs have the largest errors. However, since averaging could not halve the all bin widths perfectly, this was reflected as reduced number of code presentation in 10-bit codes and resulted in a negative INL error. Due to the rounding of the averaged values, this was observed as a larger INL error in 9-bit codes since 9-bit code's 1 LSB = 2 LSB in 10-bit code, so 8.8 LSB = 135.5 ps in 9-bit codes, -15 LSB = -115 ps in 10-bit codes. Therefore, scaling up the code reduces the quantisation errors in the averaging operation and this was reflected on the INL error.

6.4 Transfer Function of the TDC

The time is defined as a continuous variable and TDCs are used for mapping the time to discrete quantisation steps. However, due to non-linearities, these step sizes vary across the delay line and leads to unequal quantisation of time. Cumulated bin counts can be used for expressing bin widths as mentioned before. Also, by applying CDF to bin widths will reveal the transfer function of the TDC. It should be noted that when a calibration is applied, it effectively stretches the last bin of the code to 511 bins and assigns new calibrated code for each code input. This operation results in more linearised transfer function and better representation of time with 9-bit code since the last bin is stretched from 405 to 511 linearly. When the Dual Data Rate Registration is applied to the lower and upper regions of the delay line is used to represent the code, so the large bin widths are effectively split into smaller chunks. These smaller chunks result in reduced size in each step and more code presentation for inputs. Also, the code density calibration is applied to the linearised code as a part of the linearisation process.

In Figure 6.13, comparison of transfer functions obtained from the linearised & calibrated TDC, calibrated TDC and non-calibrated TDC. Linearised and only calibrated transfer function provide linear results that stretch 511 bins due to the

6.4. TRANSFER FUNCTION OF THE TDC

effect of the code density calibration. However, the non-calibrated transfer function suffers from serious non-linearities. Especially, after bin 350 this non-linearity becomes even worse and indicates severe inconsistency of delay propagation at that point. This effect was also observed earlier in Figure 6.5.

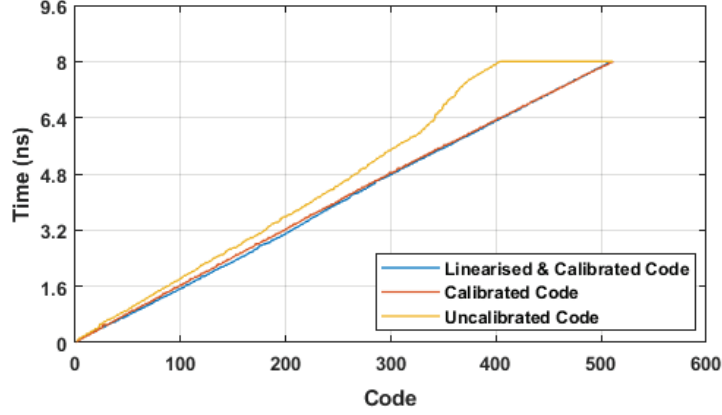


Figure 6.13: *The transfer functions for linearised, calibrated and non-calibrated codes*

When the bin widths are compared for earlier steps in the delay line as in Figure 6.14, all transfer functions are observed as much linear. Moreover, the non-calibrated transfer function seemed to provide smaller quantisation steps than the calibrated one due to stretched calibrated codes.

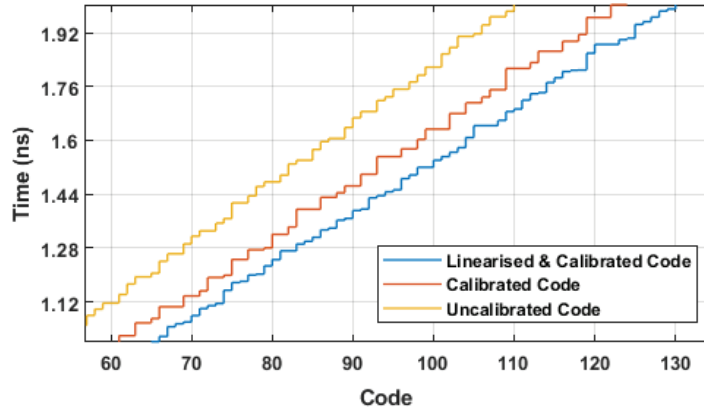


Figure 6.14: *The transfer function comparison for linearised, calibrated and non-calibrated codes for early bins*

However, around bin 350, the non-linearity observed severe as it can be seen in Figure 6.15. This figure indicates the adjacent carry chain elements are not aligned perfectly next to each other towards to end, and some large inter-slice propagation

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

delays are affecting the bin widths. Nevertheless, the calibration effectively linearised the transfer function in that region.

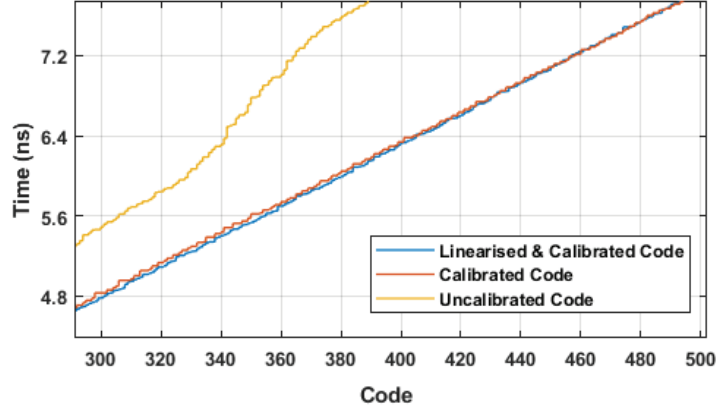


Figure 6.15: *The transfer function comparison for linearised, calibrated and non-calibrated codes for late bins*

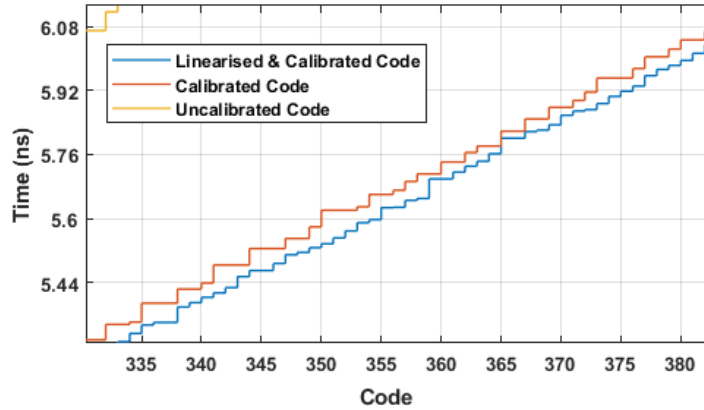


Figure 6.16: *The transfer function comparison between linearised and only calibrated codes*

When the effects of the linearisation and calibration are compared, the linearised code showed reduced sized steps in the transfer function. This effect can be seen in Figure 6.16, the linearisation mostly halved the quantisation steps since it averaged codes generated from upper and lower regions of the delay line. However, due to the quantisation errors in the averaging operation, some codes achieved results in between two quantisation steps, and for those bins, they were observed as larger bins in the transfer function of the linearised TDC. These errors can be observed between bins 355 and 360 in Figure 6.16.

6.5 Multi-Stop TDC

Multi-stop tag correlator module essentially implements a multi-stop TDC, and in order to show its functionality, a test with one start and many stops have been conducted as it was described in [14]. The test setup had two signal generators (Thurlby Thandar Instruments TGP110 10MHz) which were in-sync with each other where one of them was generating the START and while the other one was generating the STOPS. For the first two tests, the STOP signal generator was set to 1 MHz, and the START signal generator was set to $1\text{MHz} / N$, where N is the required number of STOPS. For the third test, the STOP signal generator was set to 275 kHz whilst, the START generator was set to 2.3kHz. The START signal had a slower frequency than the STOP signals in order to generate many STOPS between two START signals. Once the ΔT s were calculated, they were sent to PC for forming the histogram of the delta times. Each histogram was formed from approximately 100,000 data samples [14].

The first test was conducted for 5 STOPS. Thus, the START signal was at 200 kHz while STOPS are generated at 1 MHz. The results of this test can be seen in Figure 6.17 [14].

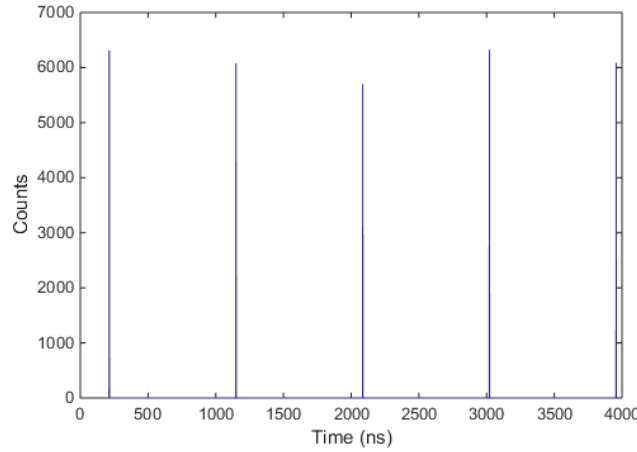


Figure 6.17: *Histogram of 5 STOPS computed by the correlator [14]*

The second test consisted of 43 STOP signals. This test also had 1 MHz clock for each STOP while START signals were given every 23.3 kHz. The results observed from this test can be seen in Figure 6.18 [14].

The third test was aimed to record 120 STOP signals. The STOP signals generated at 275 kHz while the STOP signal frequency was at 2.3 kHz. Thus, the results are obtained as in Figure 6.19 [14].

As a result of this test, a multi-stop TDC can measure time differences between a single stop and an arbitrary number of STOPS. Thus, it showed that the multi-tag

CHAPTER 6. FUNCTIONALITY, LINEARITY AND PRECISION OF THE TIME-TO-DIGITAL CONVERTER

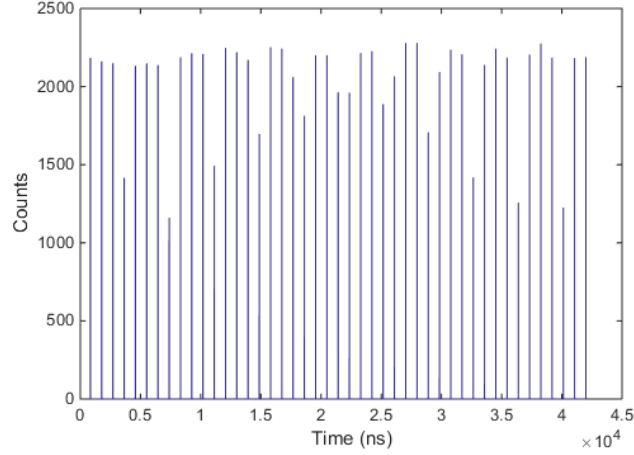


Figure 6.18: *Histogram of 43 STOPS computed by the correlator [14]*

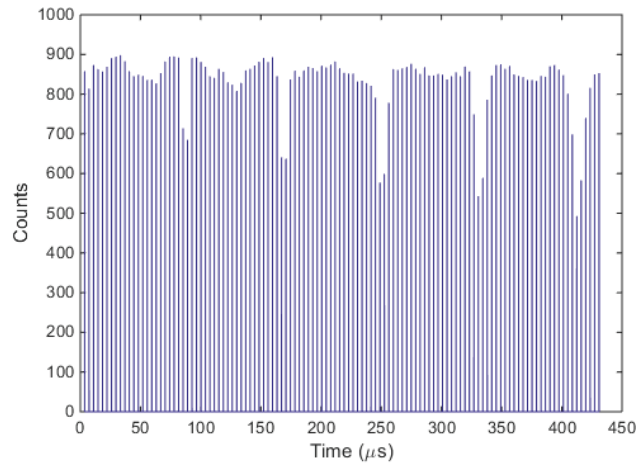


Figure 6.19: *Histogram of 120 STOPS computed by the correlator [14]*

correlator was able to operate as a multi-stop TDC successfully[14].

6.6 Summary

To sum up the work presented in this chapter, the timing jitter measurement, bin width characterisation, TDC linearity measurements, TDC transfer function analysis and multi-stop functionality tests were done on the TDC. The timing jitter measurements showed that the SSP of the TDC was observed as 8.9 ps. This precision was seen as 3.2 ps improvement after the linearisation. As the bin widths were characterised, the number of missed code in the transfer function was reduced, and around 100 more bins were observed as non-zero. As a result, a more linear transfer function was achieved,

and 1.2 LSB reduction in max DNL error and 1.8 LSB reduction in max INL were observed. Also, an improved linearity in DNL was observed as 2.9 LSB, and INL was observed as 8.8 LSB. Moreover, the TDC was successfully functioned with one START and an arbitrary number of STOP signals. Tests showed the capability of 1 START and 120 STOPs detection of the system.

In the next chapter, the results achieved from the coincidence counting experiments will be discussed.

Coincidence Counting Benchmarks and Applications

7.1 Introduction

In this chapter, the performance benchmarks of the coincidence counter system will be discussed. The performance of the coincidence counter tested from two different aspects. These aspects were the performance limits of the system and its functionality in the quantum photonics applications.

The performance limits of the coincidence counter can be tested by inputting test signals and observing the response of the system at the output. Two different tests were conducted for this purpose. In the first test, the coincidence rate as a function of the delay was tested where two identical signals with a fixed delay between them feed the system. This test should also provide the narrowest usable precise coincidence window size. The second test was conducted to measure the maximum count-rate of the system. This test was done by inputting two identical signals to different channels with fixed coincidence window and comparing the 2-fold coincidence rates between the single rates of the coincidences. This test necessarily should provide the linearity of the coincidence counter, and from the point where the coincidence rates deviate from the single rates, the highest input rate can be used on each channel was measured.

In the second part of the experiments, the coincidence counter was tested in different photonics experiments. In the first experiment, the system's performance against a popular commercial product PicoQuant PicoHarp300 in an optical setup was observed. This was essentially used for verifying the results achieved with the

CHAPTER 7. COINCIDENCE COUNTING BENCHMARKS AND APPLICATIONS

proposed system with a commercially available product. In the second test, the Rev-HOM dip interference was investigated in a quantum photonics setup. This test was fundamentally, used for verifying the system's performance with a known physics phenomena in quantum photonics experiment. The third conducted experiment was pseudo-photon-number resolving detection of a coherent state of light. This test was utilised for demonstrating the system's capability of being used in the quantum experiments by showing its performance exceeds the saturation of the photon detectors used in the experiments.

7.2 Coincidence Counting Benchmarks

7.2.1 Coincidence Rate As A Function of the Delay

To benchmark, the coincidence counter, a test of the coincidence rate as a function of the delay was conducted. In this test, two identical test signals were generated by using a signal generator (B & K Precision BK4054B), a splitter and a delay box (Ortec DB463 Delay Box) were used. The splitter split the signal into two channels of the delay box, and two identical signals with a fixed delay between them were generated. Both signals were running at 1 MHz, which implies 60000 tags were generated by each channel since the operation time of the system was set to 60 ms. The fixed delay was set around 20 ns, but due to routing and cable lengths, it was observed to be slightly more than 20 ns [3].

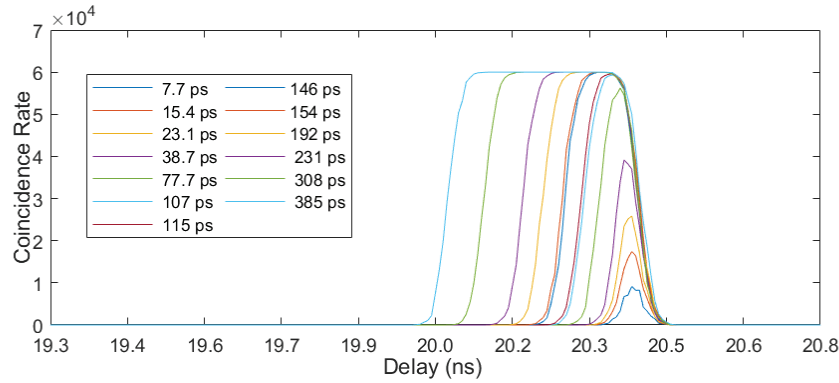


Figure 7.1: *The coincidence rate as a function of the delay with window size 380 ps - 7.7 ps*

As a result of this test, the system was proved to be working since the highest coincidence rates were observed at the same offset delay of 20.4 ns with different window sizes. However, 107 ps was the last coincidence window where all the expected coincidences were captured. Hence, 107 ps was the point where the last saturation of

7.2. COINCIDENCE COUNTING BENCHMARKS

coincidence rates was observed. In Figure 7.1, the coincidence rates for window sizes between 7.7 and 380 ps can be seen [3]. The reason results of this experiment has showed around 100 ps peak-peak jitter is due to the input time walks of the circuitry, and thus, the capabilities of the TDC scheme cannot be reflected onto coincidence counter due to this limitation.

The second delay test was conducted to show the coincidence counter's functionality with large window sizes. As it can be seen from Figure 7.2, the coincidence rates are all saturated at 60000, and the width of the pulses is roughly equal to the set coincidence window sizes. Thus, the experiment shows that the proposed system also performs as expected with larger coincidence windows [3].

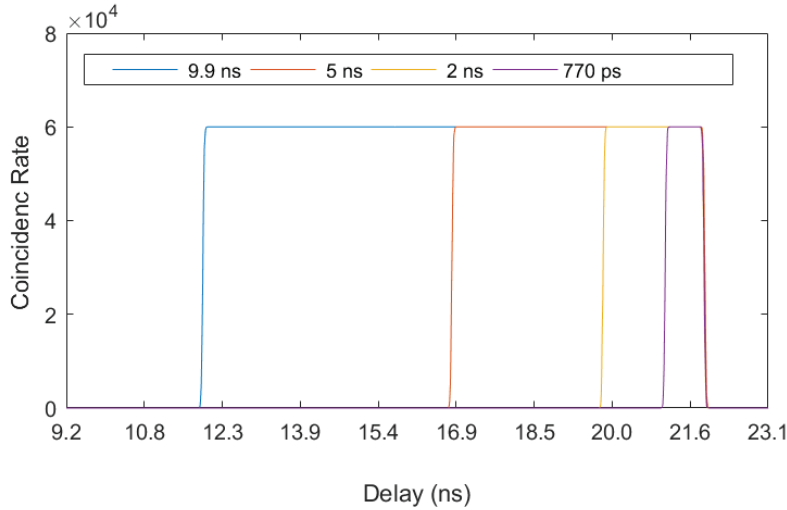


Figure 7.2: The coincidence rate as a function of the delay with window size 770 ps - 9.9ns [3]

7.2.2 Linearity of the Coincidence Counter

Also, to investigate the maximum count-rate of the system, the highest 2-fold coincidence rate that can be detected per channel was investigated. In an ideal setup, the 2-fold count-rates should be in linear relation in with the input data rate (i.e. 10 MHz = 10MCPS, 100 MHz = 100 MCPS etc.). However, due to the system's dead-time, a count-rate limit is expected. Therefore, this measurement aimed to find the last measurable count-rate where the measurements were still accurate. In order to do this, two identical signals were generated with the Xilinx DCM module varying from 3.12 MHz to 70 MHz and split into the two channels. Since the signals were the same, the 2-fold coincidence rates should have been equal to the singles rate for a channel. The window was set to 2 ns, which was large enough to compensate for any routing delays. As it was mentioned in Chapter 4, typically the clock period is the dead time of

CHAPTER 7. COINCIDENCE COUNTING BENCHMARKS AND APPLICATIONS

TDC cores, which was 8 ns in this implementation. However, the coincidence counter core requires 3 clock cycles to accumulate a coincidence into a BRAM, resulting in a maximum count-rate of 41.67 MCPS overall for the system [3].

As 3 clock cycles imply, 24 ns is the dead-time of the system. However, due to cycle-to-cycle jitter in the source repetition rate and system clock, a break-down was observed after 40 MHz. Since the system requires 24 ns ($T_{clk} * 3 = 8\text{ns} * 3$) \pm clock jitter to accumulate the coincidence, but the source is repeating at 25 ns \pm source jitter. For instance, if the most negative source jitter + the most positive clock jitter exceeds 1 ns (e.g: clock jitter = +200 ps and source jitter = -1 ns, overall 1.2 ns), then a coincidence will be lost. Thus, at 41.67 MHz around 30% of coincidences, at 45 MHz 35% of coincidences and at 50 MHz 50% of coincidences are dropped. Between 41.67 MHz source repetition rate (24 ns = 3 clock cycles) and 62.5 MHz (16 ns = 2 clock cycles), the coincidence detection efficiency drop from 100% to 50% was expected (at 2 cycles/tag, every other tag will be dropped since there will be no instances of 3 clock cycles between two tags). This expected behaviour was observed with the measurements taken after 45 MHz, and the count-rates were observed to be half of the expected theoretical rates until 60 MHz (i.e. 50 MHz = 25 MCPS, 60 MHz = 30 MCPS etc.). However, at 70 MHz (14.2 ns) since it is less than 2 clock cycles (16 ns), only 1 count could be detected for every 24 ns (i.e 70 MHz = 23.3 MCPS). These results can be seen in Fig.7.3, where the red dashed line is the measured single rates (theoretical coincidence rate), and the blue line is the measured 2-fold coincidence rates [3]. In addition, as it was discussed in section 5.3.3, the readout time also affected these measurements. Roughly around 20,000 counts per second have been lost to the readout time (0.02 MCPS).

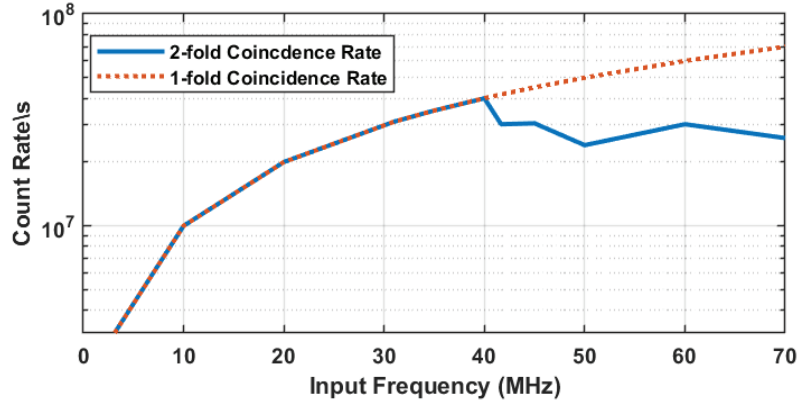


Figure 7.3: Coincidence rates (singles and 2-folds) vs input frequency [3]

7.3 Quantum Photonics Applications

7.3.1 Comparison with PicoHarp300

To verify the precision of the system, it was compared with a commercial product (PicoQuant PicoHarp300). In this test, a laser pump (Thorlabs, L405P150) at 405 nm, which was connected directly to the Single Photon Avalanche Detector (SPAD) (Excelitas, SPCM-AQ4C) with fibre connectors were utilised. The outputs of the detectors were connected to the coincidence counter's channels for characterising the offset delays between the source and the detectors. This was achieved by adding 1 LSB (7.7 ps) every 60 ms and observing the change in the coincidence rates between two channels. The detectors were also connected to the PicoHarp300 to take the same measurement.

In this experiment, the PicoHarp300 was used for measuring the time between two channels and the bin width was 4 ps. To achieve a similar output from the proposed system, the coincidence window was set to a single bin of 7.7 ps.

The PicoHarp captured the offset delay at 1.190 ns whilst, the proposed system captured the offset delay at 1.347 ns which implies both systems measured the offset delay with a difference in internal routing delays. Also, the peak of the plot was almost twice as high as the PicoHarp's since the window size was around twice as large. The FWHM of the measurement was at around 35 ps which complies with the TDC's precision.

A difference was observed in input rates between the two devices. The PicoHarp300 measured 180000 tags/s for channel 1 and 113000 tag /s for channel 2, while the proposed design's channel 1 measured 242800 tags/s and channel 2 measured 152500 tag/s. This showed that the data rate ratio between channel 1 and channel 2 was the same for both systems (1.59), and it proved that the scheme was functioning correctly. The difference between the count-rates was because the PicoHarp's measurements were limited with the data captured between START and STOP signals. Thus, once START was asserted, the next START was not processed until the next STOP. However, the proposed design did not have such a limitation, and every trigger was detected during an integration time.

7.3.2 Rev-HOM Dip Measurement in Two Single-Photon Interference

To demonstrate the suitability of the counting logic module for multi-photon experiments, the system was implemented in the detection scheme of quantum linear optical experiments. In the experiments, nanowire single-photon detectors (SNSPDs) are used, since they are very high-performance detectors, and generate apparent detection signals. In particular, they produce very low-jitter electronic signals, which are relevant in the tests given the high time precision of the electronics. The high efficiency

CHAPTER 7. COINCIDENCE COUNTING BENCHMARKS AND APPLICATIONS

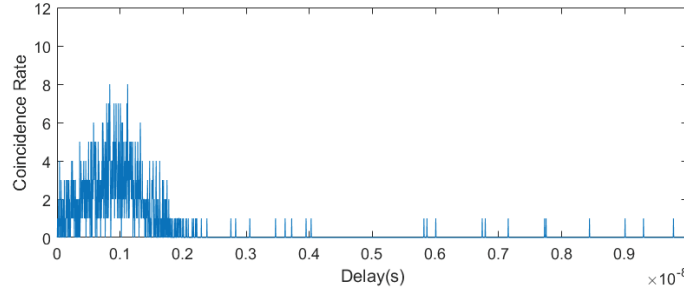


Figure 7.4: *The coincidence rate as a function of the delay measured with Picoharp300*

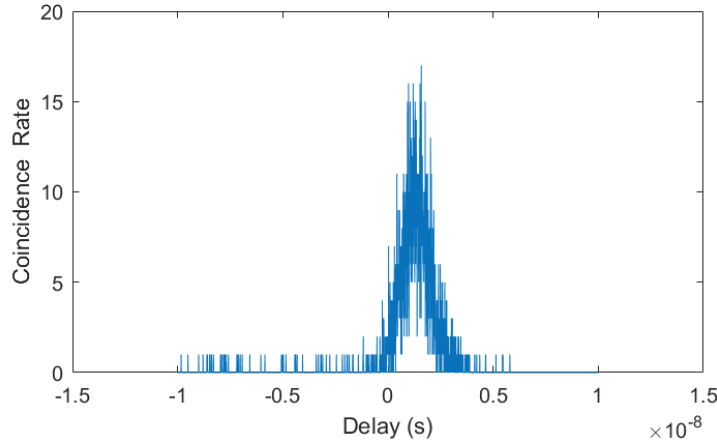


Figure 7.5: *The coincidence rate as a function of the delay measured with the coincidence system*

and low dead-time allow reaching very high count-rates in experiments, which was essential to test the counting rate of the module. More generally, they are the best commercially available single photons detectors, and thus, they are the most used detectors in multi-photon quantum optics experiments [3].

As the first benchmark, it was used for the detection of quantum interference occurring in the reversed Hong-Ou-Mandel (rev-HOM) effect [208, 209]. As represented in Figure 7.6a, the rev-HOM effect is a two-photon interference effect given by the time-reversal of the standard Hong-Ou-Mandel effect [66]. A bunched photon pair was prepared in a superposition of two-photon Fock states $(|20\rangle + |02\rangle)/\sqrt{2}$, and then an optical phase ϕ was inserted on one of the modes before injecting them into a balanced 50 : 50 beam-splitter. A pulsed laser with 2 nm bandwidth, 50 MHz repetition rate, 1 mW average power was used for this experiment. These values were chosen in order to achieve 2500 Hz coincidence rate at the peak of the fringes in Figure 7.6b, corresponding to a power of a 0.6 fW. Thus, $0.6 \text{ fW}/2500 \text{ CPS} = 0.24 \text{ aJ}$ per coincidence was obtained. This value corresponds to a 200 dB attenuation from the

7.3. QUANTUM PHOTONICS APPLICATIONS

1 mW/50 MHz = 20 nJ per laser pulse. Two single-photon detectors and the counting logic unit were finally required to measure the quantum state emerging from the interferometer. In the experiment, the photon-pair generation and the interferometer were integrated on a single silicon-photonic chip using standard silicon quantum photonics components [210, 28]: two spontaneous four-wave mixing waveguide sources were coherently pumped to generate the superposition state, and a thermal phase shifter and a 2×2 multi-mode interferometer were used to implement the phase shift and the beam-splitter, respectively [3].

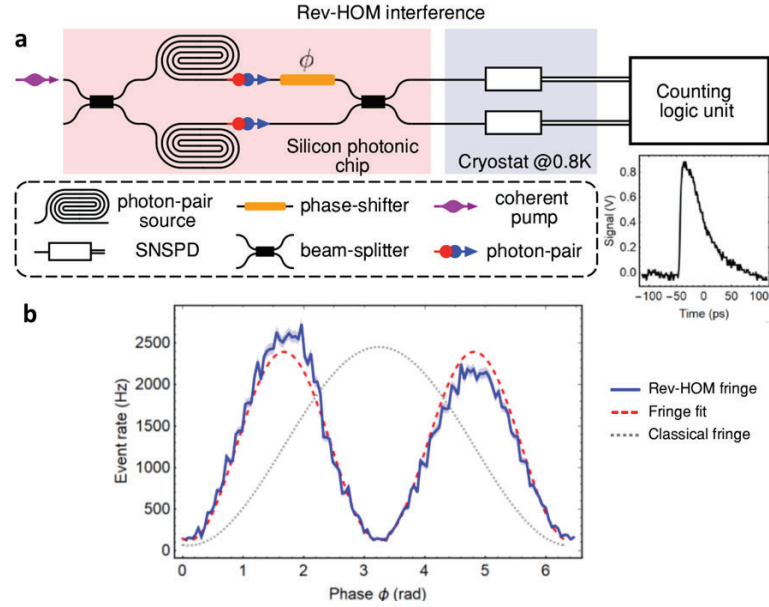


Figure 7.6: *Quantum photonics experimental benchmarks of the counting system in quantum interference experiment [3]*

In Figure 7.6 quantum photonics experimental benchmarks of the counting system can be seen. A schematic of the rev-HOM quantum interference experiment. An integrated circuit on a silicon chip is used to generate a photon-pair in a quantum superposition and perform the quantum interference, while the measurement is performed off-chip using SNSPDs and the counting system. Inset: electronic signal emitted by the detectors upon photon detection. b, Results of the rev-HOM experiment, the solid blue line represents the measured coincidence rates, which form an interference fringe with a 90% visibility as measured by the fitted curve (red dashed line). The black dashed line is the classical fringe obtained with classical light rather than single photons [3].

After the interference, photons were coupled off-chip into optical fibres and sent to the detection scheme. In the measurement, two SNSPDs [211] were employed

CHAPTER 7. COINCIDENCE COUNTING BENCHMARKS AND APPLICATIONS

(PhotonSpotTM) with approximately 85% efficiency, 100 Hz dark counts, 50 ns dead-time and 70 ps internal jitter. After the amplification, the electronic signal emitted from each detector upon photon detection was given by an exponential decay shape with a peak of approximately 0.8 V (see inset of Figure 7.6a). The electronic signals from the two SNSPDs were sent into two input channels of the counting logic unit, set to a threshold of 0.1 V, which then counts the simultaneous events within a coincidence window of 2 ns. To observe the rev-HOM quantum interference, The coincidence events rate were continuously monitored while scanning the phase between 0 and 2π . The results are shown in Figure 7.6b. The measured rev-HOM quantum interference fringe, which presents the typical doubled frequency when scanning ϕ respect to the classical fringe [208], was observed with visibility of 90%, consistent with previous measurements on the same device [3]. The visibility was also affected by the readout time of 20.4 μ s between each measurements and with an implementation a circular buffer like in [155] can slightly improve the results.

7.3.3 Pseudo-photon-number Resolving Detection of a Coherent State of Light

The capability of the device to support high count-rate for multi-photon experiments by using it to perform pseudo-photon-number resolving detection of a coherent state of light with up to eight-photon resolution. This was implemented by injecting weak-coherent pulses into a 1×8 fibre beam-splitter, which probabilistically splits the photons of the coherent state of light into the 8 output channels measured via 8 SNSPDs see Figure 7.8a.

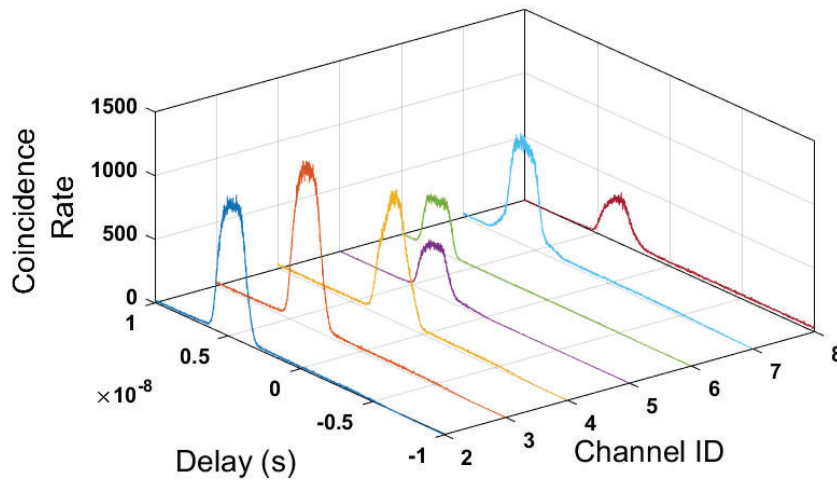


Figure 7.7: The detectors' delay calibration with respect to the reference channel (channel 1) [3]

7.3. QUANTUM PHOTONICS APPLICATIONS

In Figure 7.8a, the schematic of the pseudo-photon-number resolving measurement on a coherent state. b. Results of the pseudo-photon-number resolving experiment for different amplitudes of the weak-coherent state. Marks with different colours refer to data for different n -fold coincidence events, up to $n = 8$ photons. Dashed lines were obtained by fitting the data to the expected rates from a coherent state, assuming uniform channel and detector efficiencies. The shaded area represents the region where the SNSPDs start to saturate. Error bars were at one standard deviation confidence intervals, calculated from Poissonian counting statistics (not shown when smaller than markers).

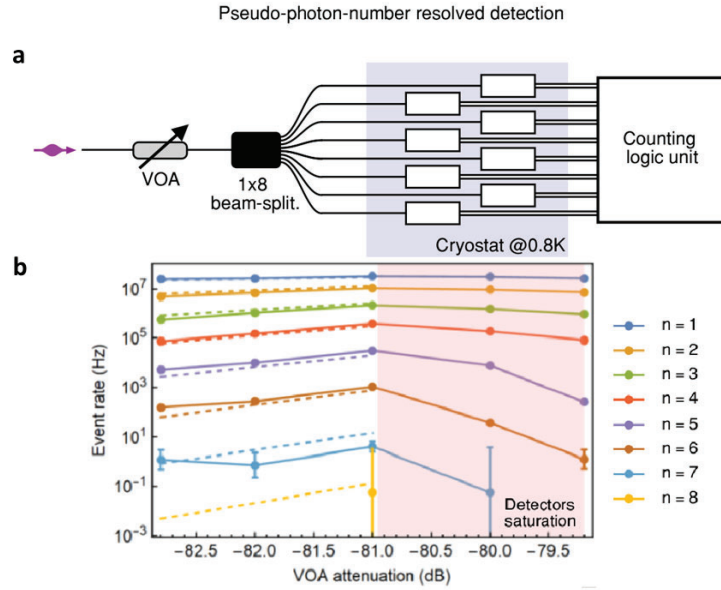


Figure 7.8: *Quantum photonics experimental benchmarks of the coincidence counting system in a pseudo-photon-number resolving detection of a coherent state of light [3]*

Correlation measurements between the detection events from the 8 SNSPDs were performed using the 8 input channels of the counting logic unit. The detector delays were calibrated with respect to the channel 1 by adding delays to other channels and sweeping results within -10 ns to 10 ns range for measuring the 2-fold coincidences between channel 1 and the others. This can be seen in Figure 7.7.

A Variable Optical Attenuator (VOA) was used to tune the amplitude of the coherent state, which allowed the output photon distributions for various amplitudes were to be measured. Results are shown in Figure 7.8c. The counting logic unit was able to support more than 50 MHz singles event rate over all channels and was used to measure up to 8-photon coincidences. The measured count-rates were limited only by the saturation level of the SNSPDs, which started to saturate at a few MHz counts,

CHAPTER 7. COINCIDENCE COUNTING BENCHMARKS AND APPLICATIONS

rather than the capability of the counting electronics. This test demonstrated that the processing capability of the counting logic unit surpassed the detection rate capacity of the single-photon detectors, showing its suitability for many-photon experiments where high count-rates over many channels are required [18].

7.4 Summary

To sum up this chapter, the proposed coincidence counting system was tested in different scenarios. These scenarios aimed to verify and show the limits and functionality of the design. To demonstrate the coincidence detection limit of the system coincidence rate as a function delay test was conducted to measure the fixed delay between two channels. This also provided the minimum accurate coincidence window size. As a result of this test, the minimum precise window size was measured as 107 ps and this showed around 100 ps peak-to-peak jitter. The jitter being larger than the TDC's SSP (6) was the time walks of the input circuitry. The second performance limit test was input rate versus the coincidence rate. This test primarily utilised for finding the highest input rate of coincidence counter that can be used. As a result, 40 MHz input was the last usable input frequency which corresponds to 320 MCPS for 8 channels.

Also, some tests were carried out to verify the system's outputs and its the functionality in an actual photonics setup. To verify its functionality in photonics setup, the first tests conducted was its comparison with PicoHarp300. As a result of this experiment, the coincidence rate as a function of delay test conducted on both systems achieved matching results. The second test was conducted to observe the phenomena called the Hong-ou-Mandel effect. By observing the change in the Rev-HOM dip effect between channels when the phase varied between 0 to 2π , the system's correct functionality was demonstrated in a quantum photonics experiment. The third test was the experiment of pseudo-photon-number resolving detection of a coherent state of light. This test was conducted to see whether the theoretical high count-rate of the system would also provide a high count-rate and suitability in an actual experiment. Thus, it would perform beyond the point where the SNSPDs saturate. SNSPDs' saturation was clearly observed as results of this experiment as the attenuation was dropped -81 dBs with around a total of 50 Million time tags were generated.

With this chapter, the results obtained from the coincidence counting system concludes, and the thesis will continue with the final chapter for conclusions.

CHAPTER 8

Conclusions

8.1 Conclusion

To conclude, in this thesis the research done on developing a precise high count-rate multi-channel coincidence counting instrument was presented, which was unique in a way that it integrates the time-tagging with coincidence counting operation in the same FPGA chip. This provided a very high count-rate with precision and multi-channel operation than any other available implementation (see Table 8.1). The thesis highlighted problems and challenges required to achieve these goals, and these challenges were listed as the timing precision, data throughput and maintenance of real-time. The provided coincidence counting scheme aimed to provide solutions to these problems. In order to present the research outcomes, first, the fundamentals behind the research were provided as a research review. This research review included, LiDAR techniques, TDC methods, coincidence counting methods and quantum photonics applications, where the system was tested in. In the chapter following the research review, the legacy coincidence counting implementations developed by the author were discussed. After limitations of the legacy implementations discussed, the system's benchmarks were presented. The timing of the system achieved 8.9 ps single-shot precision, while the bin width was 7.7 ps. The largest DNL error was measured as the 2.9 LSB, and largest INL error was 8.8 LSB. The TDC method used for this implementation was called as the Dual Data-Rate Registration TDC. The system clock used for this operation was 125 MHz, and so, the theoretical dead-time was 8 ns. The new Dual Data-Rate Registration scheme registers the trigger with both clock edges, average them later to calibrate the final code by using the code density testing. This linearisation method

CHAPTER 8. CONCLUSIONS

Performance Parameters	This Work	Swabian Instruments Time Tagger Ultra [21]	IDQuantique ID900 [30]	PicoQuant Hydra-harp400 [20]	[31]	[32]	[33]
Total Count Rate	320 MCPS	65 MCPS	100 MCPS (13 ps resolution), 400 MCPS (100 ps resolution)	40 MCPS	72 MCPS	0.5 MCPS	32 MCPS
Resolution	7.7 ps	3 ps	13 ps at hi-res, 100 ps at low-res	1 ps	68.3 ps	390 ps	0.7 ns
Single-Shot Precision	8.9 ps	10 ps	8 ps	8.5 ps	N/A	N/A	0.7 ns
Slices	11,636 (50%)	N/A	N/A	N/A	66%	2257 (99%)	N/A
Registers	24926 (13%)	N/A	N/A	N/A	94%	6051 (33%)	N/A
I/O	85 (100%)	N/A	N/A	N/A	89%	153 (65%)	N/A
Channel No:	8	14	4	8	12	32	64
DNL	2.9 LSB	N/A	N/A	2% peak	N/A	1 LSB	N/A
INL	8.8 LSB	N/A	N/A	N/A	N/A	N/A	N/A

Table 8.1: *Performance parameters of the proposed coincidence counting system and existing TDC based coincidence counters*

achieved improvements of 1.2 LSB in Max DNL error, 1.8 LSB INL error, 10 ps in FWHM and 3.2 ps in RMS resolution. Thus, this TDC technique was used to implement a coincidence counting method, which was based on forward looking tag-difference and multi-stop LiDAR methods. The system achieved 40 MCPS per and channel 320 MCPS for the entire 8-channel system. The smallest window size was 7.7 ps, and however, the smallest precise window was measured as 107 ps. Around 100 ps peak-to-peak jitter resulted in the measurements was caused by a time of walk of input circuitry. This system was utilised in different photonics setups. In the first test, it was tested against PicoHarp300, and matching results were obtained from both devices. In the second test, the system was tested in the measurement of Rev-HOM dip effect, and this phenomenon was successfully observed. Finally, it was tested with a coherent state of the light experiment, and it showed functionality beyond the saturation of the detectors.

The main limitation of the developed instrumentation was a requirement for large

memory in an FPGA, and, as the number of channels increases, the memory needs an exponential rise. Typically, a need for large memory requirements results in the placement of more RAM blocks inside the FPGA fabric, which makes the design harder to route without violating the timing constraints. In current Xilinx Spartan 6 architecture implementation of 8 channels was the limit, and, for more channels, much larger FPGA architectures should be considered. Furthermore, although, TDC provides 8.9 ps SSP, the actual precision of the coincidence counting was limited by the time of walk of the input circuitry. Therefore, improvements in the timing instrument breakout board and using faster FPGA chips should be considered for improving the jitter affecting the coincidence counting operation. Also, a slight readout dead-time of 20.4 μ s affecting the results, for the future a cyclic buffer for the readout logic should be considered to eliminate this time.

As final words, this research provided an effective method for a precise multi-channel high count-rate coincidence counting operation. It has shown to provide a desired coincidence counting functionality for quantum photonics experiment, where increasing complexity hinders the operation of current instruments. In the future, the research can be extended by developing schemes which will increase the number of operational channels, count-rate by implementing a dual-data rate TDC and improving the linearity and precision by introducing multi-phase registration methods. For the future works the section 8.3 should be referred. In the next section chapter summaries will be presented.

8.2 Summary

In this thesis, the research done on high count-rate precise multi-channel coincidence was presented in 7 chapters excluding Conclusion. In this section a brief summary for each chapter will be provided.

In Chapter 1, an introduction to the timing measurement tools such as TDCs and coincidence counters were provided. Constraints which are needed to be overcome in order to implement such scheme were also discussed in this chapter. The main constraints faced while implementing such a scheme included the timing precision, maintenance of real-time operation and data transfer throughput. Therefore, the proposed scheme needed to address these challenges. Research contributions are also summarised in this section. In TDC side of the contributions include the development of 8.9 ps (LSB 7.7 ps) single-shot precision TDC design using the dual-data rate registration in combination with the code density calibration method. With the usage of the Dual Data Rate Registration 1.2 LSB in Max DNL error, 1.8 LSB INL, 10 ps in FWHM and 3.2 ps in SSP improvements were achieved. Contributions done on coincidence counters research can be summarised as a development of a coincidence counting scheme using a time tag-based multi-stop LiDAR correlation to implement

CHAPTER 8. CONCLUSIONS

a multi-channel labelling fast real-time coincidence counter. This scheme integrates a TDC and coincidence counter into the same FPGA fabric to solve the data transfer throughput problem. Also, this scheme operates a coincidence counting and TDC concurrently for each channel to achieve an optimised real-time operation. This scheme managed to measure all the coincidences with 107 ps window sizes and window sizes as small as 7.7 ps could be used. A count-rate of 40 MCPS per channel and 320 MCPS for the entire 8 channel system was achieved.

Chapter 2 presents the research review done on the ToF methods, quantum information concepts such as quantum interference, TDCs and coincidence counters. ToF methods focused on modulation, pulse LiDAR variations and theory behind the concepts. A brief theory about the quantum information theory, HOM-dip effect and quantum interference were provided. These photonics concepts were necessary to understand the applications where the proposed scheme was used. Research review also included TDC methods implemented by using various technologies include FPGA, ASIC or Analogue Circuits. Typically between SSPs of 770 fs - 400 ps were achieved. Coincidence counter review included implementations using combinatorial logic, analogue circuits, time tag-based methods, SFQs circuits and some specific coincidence counting methods for certain applications such as nuclear sciences.

Chapter 3 presents the legacy of the coincidence counting system, where obsolete time tag-based coincidence counting systems designed by the author were discussed. These systems were the software-based real-time coincidence system, FPGA based backward looking tag difference coincidence counting system and tag serialising FPGA based forward looking tag difference coincidence counting system. The software implementation managed to prove the concept of the forward looking tag difference method, and however, it could not achieve high count-rate due to the USB throughput between the TDC and coincidence counter. The second system was the first successful integration of the coincidence counting and TDC into the same FPGA. This implementation used a RAM block as a buffer to search for coincidences, which resulted in high jitter at the results. The third approach was very similar to the final system apart from using a tag serialiser to connect the TDC to the coincidence counter. Therefore, the count-rate was limited by the tag serialiser. However, the third implementation successfully used the multi-stop LiDAR method to implement a real-time coincidence counter.

Chapter 4 presents the implementation details of the proposed TDC scheme. This includes how the carry chain's state is captured, and code is generated, the details of the calibration module and Dual Data Rate Registration. A 511-bin delay line was implemented using the Spartan 6 LX150 FPGA located on the Opal Kelly XEM6310. The system clock was set to 125 MHz (8 ns). The double registration was implemented by capturing the state of the same delay line with both edges of the clock, and these codes were averaged. The calibration block used the code density testing method to calibrate the averaged codes, and the final code is scaled up to 1023-bins (10-bit) in

the calibration block. Also, the LSB resolution of the TDC was set to 7.7 ps.

Chapter 5 presents the implementation details of the coincidence counter scheme. The coincidence counter implementation details include the multi-tag correlator, coincidence detector and coincidence counter are presented in this chapter. Also, the hardware and software of the implementation documented in this chapter. The multi-tag correlator continuously subtracts time differences between the START and many STOP signals from the perspective of a channel, which essentially uses the multi-stop LIDAR correlation. These time differences are used for the coincidence address formation in the coincidence detection module. The coincidence addresses are used as indices of a RAM block in the coincidence counter block and incremented each time a coincidence address is passed to the coincidence counter. It should be noted that each channel executes these operations in parallel. For this research, a TDC breakout board was designed for the Opal Kelly board which was developed in the University of Bristol Photonics group. This board accommodated a low jitter clock, jitter attenuators and 8 I/O ports. The software interface was implemented by using the Opal Kelly API to interface the USB 3.0. Also, the software includes post-processing to fix duplicates and missed coincidence counts.

Chapter 6 discusses the performance of the TDC in terms of functionality, linearity and precision. Functionality was discussed in terms of multi-stop operation. The design successfully managed to capture 120 STOP signals between two START signals. The linearity of the TDC was discussed from 3 parameters; bin widths, DNL and INL. The bin widths of the design were compared before and after the calibration and linearisation. The improvement in max INL error after calibration achieved as a drop from -30 LSB to 10.8 LSB. The maximum DNL error after the linearisation which was dropped from 4.1 LSB to 2.9 LSB, and generally the DNL was smaller across the delay line. After the linearisation, INL values were observed to be much closer to zero, and a maximum of 8 LSB INL error was achieved. In addition, the SSP of the TDC compared before and after the calibration and linearisation. Before the calibration, the single-shot precision was measured as 22.6 ps, after 12.1 ps and after the linearisation, it was observed as 8.9 ps RMS. These results showed clearly the effects of the calibration and linearisation on the TDC.

In Chapter 7, the benchmarks obtained from the coincidence counter implementation was discussed. These tests demonstrated the functionality of the system in quantum photonics applications, the limits of the instrument and its verification. To test the design's functionality, two quantum photonics experiments were used, which were the detection of quantum interference occurring with the rev-HOM dip effect and pseudo-photon-number resolving detection of a coherent state of light. The rev-HOM effect was successfully observed with the coincidence measurement as the phase between the channels changed between 0 to 2π . In the pseudo-photon-number resolving detection of a coherent state of light, multi-fold coincidences rates are measured and

coincidences rates were observed as within 1σ error bars with an ideal. Also, the coincidence counter showed the functionality of working beyond the point of detector saturation. When the limits of the system tested, the highest count-rate of a single channel observed as 40 MCPS, and for the entire system, it was 320 MCPS. The smallest window size that could capture all the coincidences precisely was observed as 107 ps. However, the smallest window could be set to 7.7 ps. The reason of timing jitter being larger than the TDC's 8.9 ps SSP was the time of walk of the input circuitry. Also, the system's results were verified by using PicoQuant PicoHarp300, and both devices are used for measuring the fixed delay in an optical setup. The results obtained from both devices were agreeing with each other. Thus, the proposed system's functionality has been verified.

8.3 Future Works

The future works of the research focus on improving the ideas represented in this thesis. These ideas include improving the precision by improving the linearity of the TDC, improving the data-rate of the coincidence counting and increasing the number of channels that the system can handle. In this section, possible improvements for future works will be discussed.

8.3.1 Improving Precision and Linearity of the TDC

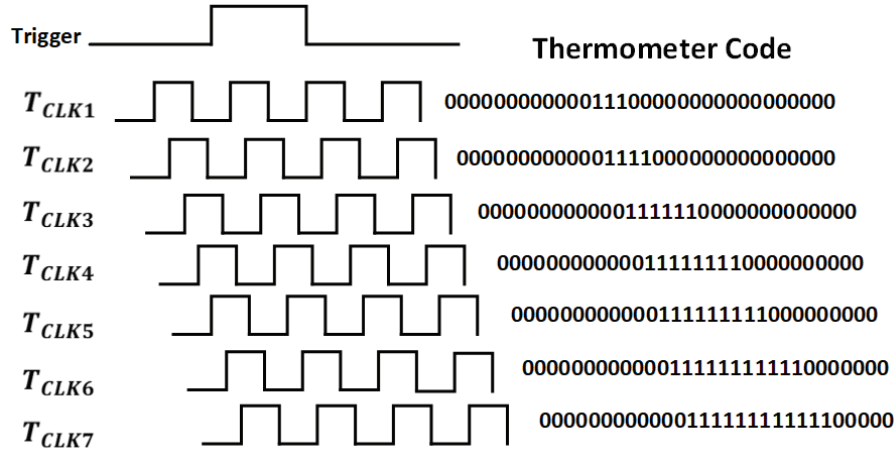


Figure 8.1: *Multi-phase registration TDC Waveforms*

Concepts of linearity and precision cannot be thought independent from each other as discussed before. The improvement in linearity will also benefit the SSP of the TDC, and therefore, the Dual Data Rate Registration method was invented in this research and improvements in both precision and linearity were observed. The Dual Data Rate

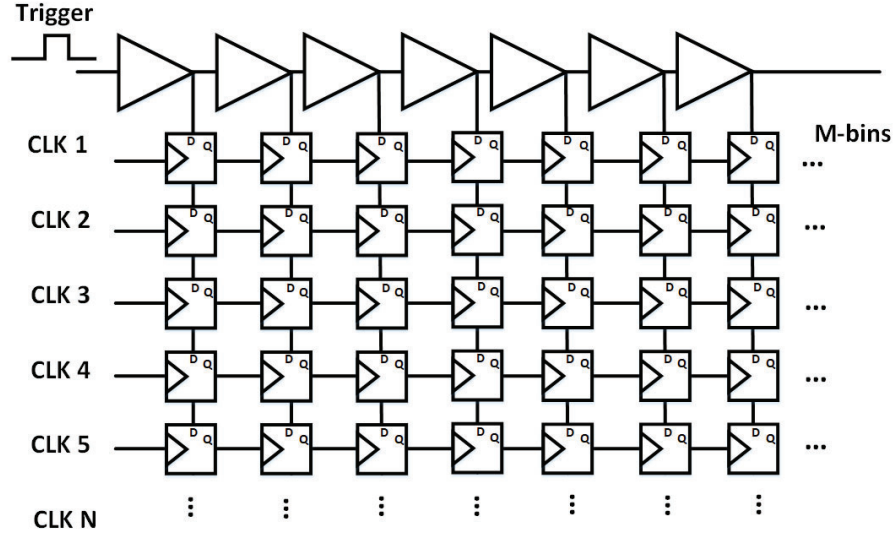


Figure 8.2: Multi-phase registration TDC architecture

Registration essentially uses two different phased clock to achieve this. Hence, to improve this method, multiple phased clocks can be used like in multi-chain averaging TDC. As the paper [159] proposes a single shot precision around 1-2 ps range could be possible to achieve when 8 different clock phases were used.

In order to this, the clock region detection is needed to be implemented for all multi-phase clocks. This can be done by dividing the clock period into N pieces, where the N is the number of clock phases. Therefore M -binned delay line is needed to be divided into M/N regions, and based on the region that trigger is, the code is added. For each clock phase regions $T_{fine} + (M/N) \times k$, where the k is the region between 0 to N , The phase of the clock capturing the fine can be represented as $T_{clk\phi} = T_{clk}/\phi$. The explanatory diagram can be seen in Figure 8.1.

Each $T_{clk\phi}$ captures the state of the delay line with D-type flip-flops, and after the correction added based on the trigger signal's detected region, M number of different codes are averaged for dividing the code to M . Thus, additional M -bits are added to the fine code. Therefore, the new LSB resolution become $T_{clk}/2^{M+\log(N+1)-1}$. For instance, for 511 bins and 8 phased clock would provide LSB of $8 \text{ ns}/2^{15} = 0.24 \text{ ps}$. Illustration of the components of the diagram can be seen in Figure 8.2.

8.3.2 Dual Data Rate TDC

Dual Data Rate Registration method showed how both clock edges could be used in a TDC scheme. Thus, this could be improved in order to implement a TDC scheme which uses both clock edges to operate. Dual Data Rate TDC can be implemented by registering the delay line with a rising edge like normal, but by modifying the priority

encoder to encode two thermometer codes from a single delay line. Two-code encoding is only enabled when two trigger signals were detected. Therefore, this method can effectively halve the dead-time and achieved a double data rate operation of the TDC channel.

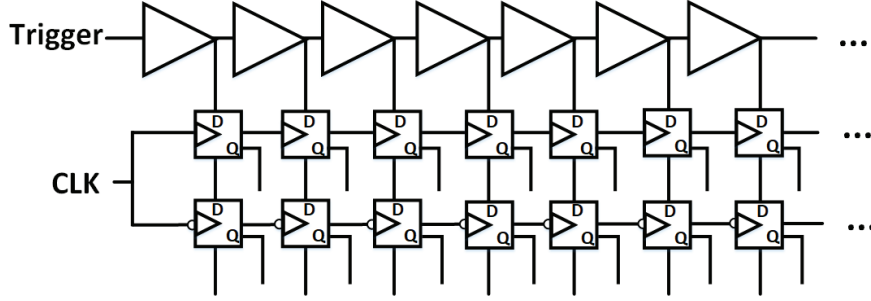


Figure 8.3: *Dual Data Rate TDC architecture*

The way to capture double triggers is using a grey code counter method mentioned earlier. For instance, if two bits of the counter is changed, this would indicate double trigger signals. Whenever double triggers are detected, the priority encoder decodes two separate thermometer codes, which are separated by trailing zeros. This can be done by detecting the first '1' to '0' transition and biggest bin with '1'. Therefore, whenever there are two trigger signals detected, the first fine code is the '1' to '0' transition and the largest '1' bin is the second fine tag. The first trigger uses the first half of the clock period to quantise the time, while the second half of the period is used for the second trigger. This operation is done by subtracting the first fine tag from the coarse counter corresponding to the rising edge, while for the second fine tag it is added to the coarse counter at the rising edge. This architecture can be seen in Figure 8.3 and its waveform Figure 8.4.

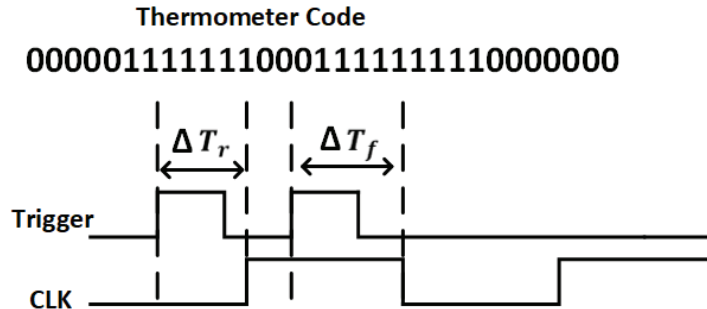


Figure 8.4: *Dual Data Rate TDC Waveform*

8.3.3 Improving Coincidence Counter Channel Number

The final crucial future work of this research include researching methods to improve the number of coincidence counter channels. The main limitation of this research is the need for an exponential increase in the amount of memory needed as it was mentioned before. Therefore, one way to get around this problem is a migration of implementation to a newer FPGA chip, but this is still not a valid option to achieve a very high channel number due to the available FPGA options. One way to address this problem can be compressing the size of the counter located inside the RAM block. For example, the counter address known to be storing very few addresses can be tried to be combined, and as a result, the memory requirement can be reduced.

Another approach can be using an arbitrary number of channels but only counting their N-fold patterns. This can drop the logic utilisation of single channel from 2^N to $\binom{N}{M}$, where N is the total number of channels, and M is the biggest fold patterns stored. This method will not be able to capture every coincidence between channels, but it will improve the detection rate of some of the larger-fold coincidences. This is due to the increase in channel numbers, which results in higher count-rate in the total system also, if the memory required for capturing every 16 channel pattern is 65536 while $\binom{16}{8}$ is 12870, which is almost 5 times less.

Bibliography

- [1] S. Tancock, E. Arabul, and N. Dahnoun, "A review of new time-to-digital conversion techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, pp. 1–1, 08 2019.
- [2] OpalKelly, "Xem6310." <https://opalKelly.com/products/xem6310/>.
[online] Accessed on 10/11/2019.
- [3] E. Arabul, S. Paesani, S. Tancock, J. Rarity, and N. Dahnoun, "A precise high count-rate fpga based multi-channel coincidence counting system for quantum photonics applications," *IEEE Photonics Journal*, vol. 12, no. 2, pp. 1–14, 2020.
- [4] C. Y. CHEN, *A Sub-centimeter Ranging Precision LiDAR Sensor Prototype based on ILO-TDC*.
Thesis, 2016.
- [5] W. K. Hamoudi and N. M. G. Al-Saidi, "Information security-based nano- and bio-cryptography," *Advances in Information Security, Privacy, and Ethics Multidisciplinary Perspectives in Cryptology and Information Security*, p. 179–199.
- [6] P. Papanastasiou, C. Lupo, C. Weedbrook, and S. Pirandola, "Quantum key distribution with phase-encoded coherent states: Asymptotic security analysis in thermal-loss channels," *Physical Review A*, vol. 98, no. 1, 2018.
- [7] E. Arabul, J. Rarity, and N. Dahnoun, "Fpga based fast integrated real-time multi coincidence counter using a time-to-digital converter," *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, 2018.
- [8] Y. Wang and C. Liu, "A nonlinearity minimization-oriented resource-saving time-to-digital converter implemented in a 28 nm xilinx fpga," *IEEE Transactions on Nuclear Science*, vol. 62, pp. 1–1, 10 2015.
- [9] E. Arabul, *PhD Second Year Report*.
June 2017.

BIBLIOGRAPHY

- [10] Y. Liu, U. Vollenbruch, Y. Chen, C. Wicpalek, L. Maurer, Z. Boos, and R. Weigel, "Multi-stage pulse shrinking time-to-digital converter for time interval measurements," in *2007 European Microwave Integrated Circuit Conference*, pp. 267–270, Oct 2007.
- [11] E. Arabul, *PhD First Year Report*.
Oct 2016.
- [12] E. Arabul, *Real Time Multi-Coincidence Counter Using Time to Digital Converter*.
Thesis, University of Bristol, 2015.
- [13] L. Bonolis, "Walther bothe and bruno rossi: The birth and development of coincidence methods in cosmic-ray physics," *American Journal of Physics*, vol. 79, no. 11, p. 1133–1150, 2011.
- [14] E. Arabul, A. Girach, J. Rarity, and N. Dahnoun, "Precise multi-channel timing analysis system for multi-stop lidar correlation," *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2017.
- [15] A. Aspuru-Guzik and P. Walther, "Photonic quantum simulators," *Nature Physics*, vol. 8, no. 4, p. 285, 2012.
- [16] S. Pirandola, U. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, *et al.*, "Advances in quantum cryptography," *arXiv preprint arXiv:1906.01645*, 2019.
- [17] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 333–342, ACM, 2011.
- [18] S. Paesani, Y. Ding, R. Santagati, L. Chakhmakhchyan, C. Vigliar, K. Rottwitt, L. K. Oxenlowe, J. Wang, M. G. Thompson, and A. Laing, "Generation and sampling of quantum states of light in a silicon chip," *Nature Physics*, vol. 15, pp. 925–929, 2019.
- [19] D. J. Brod, E. F. Galvão, A. Crespi, R. Osellame, N. Spagnolo, and F. Sciarrino, "Photonic implementation of boson sampling: a review," *Advanced Photonics*, vol. 1, p. 034001, 3 2019.
- [20] PicoQuant, "Home." <https://www.picoquant.com/products/category/tcspc-and-time-tagging-modules>.
[online] Accessed on 23/09/2019.

- [21] S. Instruments, “Time tagger series.”
<https://www.swabianinstruments.com/static/downloads/TimeTagger.pdf>.
[online] Accessed on 1/08/2020.
- [22] IDQuantique, “Id281 superconducting nanowire.”
<https://www.idquantique.com/quantum-sensing/products/id281/>.
[online] Accessed on 23/07/2020.
- [23] B. K. Das, “Positron emission tomography: An overview,” *Positron Emission Tomography*, p. 1–6, 2014.
- [24] M. Technologies, “Neutron detection and counting.”
<https://www.mirion.com/learning-center/nuclear-measurement-fundamental-principles/nuclear-measurement-fundamental-principle-neutron-detection-and-counting>.
[online] Accessed on 23/09/2019.
- [25] A. O. C. Davis, P. M. Saulnier, M. Karpiński, and B. J. Smith, “Pulsed single-photon spectrometer by frequency-to-time mapping using chirped fiber bragg gratings,” *Optics Express*, vol. 25, no. 11, p. 12804, 2017.
- [26] the Editors of Encyclopaedia Britannica, “Coincidence counting.”
<https://www.britannica.com/science/coincidence-counting>.
[online] Accessed on 26/11/2019.
- [27] Gard, B. T., K. R., Olson, J. P., Rohde, P. P., and J. P., “An introduction to boson-sampling,” 07 2014.
- [28] J. Wang, S. Paesani, Y. Ding, R. Santagati, P. Skrzypczyk, A. Salavrakos, J. Tura, R. Augusiak, L. Mančinska, D. Bacco, *et al.*, “Multidimensional quantum entanglement with large-scale integrated optics,” *Science*, vol. 260, p. 285, 2018.
- [29] P. Eraerds, M. Legre, A. Rochas, H. Zbinden, and N. Gisin, “Sipm for fast photon-counting and multiphoton detection,” *Optics Express*, vol. 15, no. 22, p. 14539, 2007.
- [30] IDQuantique, “Id900 time controller.” <https://www.idquantique.com/single-photon-systems/products/id900-time-controller/>.
[online] Accessed on 23/09/2019.
- [31] C. Wang, H. Li, Y. Zhang, H. Baghaei, R. Ramirez, S. Liu, S. An, and W.-H. Wong, “A real time coincidence system for high count-rate tof or non-tof pet cameras using hybrid method combining and-logic and time-mark technology,” *2009 16th IEEE-NPSS Real Time Conference*, 2009.

BIBLIOGRAPHY

- [32] W. Li, Y. Hu, H.-S. Zhong, Y.-F. Wang, X.-L. Wang, C.-Z. Peng, and X. Jiang, "Time-tagged coincidence counting unit for large-scale photonic quantum computing," *Review of Scientific Instruments*, vol. 89, no. 10, p. 103113, 2018.
- [33] M.-A. Tetrault, M. Lepage, N. Viscogliosi, F. Belanger, J. Cadorette, C. Pepin, R. Fontaine, and R. Lecomte, "Real-time coincidence detection system for digital high resolution apd-based animal pet scanner," *IEEE Nuclear Science Symposium Conference Record*, 2005.
- [34] H. Zhang, L. Xiao, B. Luo, J. Guo, L. Zhang, and J. Xie, "The potential and challenges of time-resolved single-photon detection based on current-carrying superconducting nanowires," *Journal of Physics D: Applied Physics*, vol. 53, no. 1, p. 013001, 2019.
- [35] L. You, "Superconducting nanowire single-photon detectors for quantum information," *Nanophotonics*, vol. 9, no. 9, pp. 2673 – 2692, 2020.
- [36] A. K. Gupta, R. S. Prasad, L. Srivani, D. T. Murthy, B. K. Panigrahi, and G. Raghavan, "Design and development of flexible and low-cost coincidence counting unit," *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2018.
- [37] A. M. Helmenstine, "What is the difference between accuracy and precision?." <https://www.thoughtco.com/difference-between-accuracy-and-precision-609328>, Sep 2019.
- [38] O. Cameron, "An introduction to lidar: The key self-driving car sensor." <https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff>, Sep 2017.
[online] Accessed on 23/09/2019.
- [39] LeddarTech, "Automotive." <https://leddartech.com/market/automotive/>.
[online] Accessed on 23/09/2019.
- [40] GeoSLAM, "About geoslam." <https://geoslam.com/about-geoslam/>.
[online] Accessed on 23/09/2019.
- [41] G. Drone, "Application of lidar in civil engineering." <http://lidarradar.com/apps/application-of-lidar-in-civil-engineering>.
[online] Accessed on 23/09/2019.

- [42] D. Schmidt, “Mine vision systems, peck tech joining forces.”
<https://www.miningmagazine.com/technology-innovation/news/1364623/mine-vision-systems-peck-tech-joining-forces>,
Sep 2019.
[online] Accessed on 23/09/2019.
- [43] U. University, “Time-of-flight measurements in quantum mechanics.”
<https://www.uni-ulm.de/en/nawi/institute-of-quantum-physics/research/research-areas/foundations-of-quantum-mechanics/time-of-flight-measurements-in-quantum-mechanics/>, Jan
2018.
[online] Accessed on 23/09/2019.
- [44] Presagis, “Lidar.” <https://www.presagis.com/en/glossary/detail/lidar/>.
[online] Accessed on 10/11/2019.
- [45] R. Whyte, L. Streeter, M. J. Cree, and A. A. Dorrington, “Application of lidar techniques to time-of-flight range imaging,” *Appl. Opt.*, vol. 54, pp. 9654–9664, Nov 2015.
- [46] C. Wolff, “Radar basics.” <http://www.radartutorial.eu/02.basics/FrequencyModulatedContinuousWaveRadar.en.html>.
[online] Accessed on 23/09/2019.
- [47] F. R. Giorgetta, E. Baumann, K. Knabe, I. Coddington, and N. R. Newbury, “High-resolution ranging of a diffuse target at sub-millisecond intervals with a calibrated fmcw lidar,” in *2012 Conference on Lasers and Electro-Optics (CLEO)*, pp. 1–2, May 2012.
- [48] B. Behroozpour, P. A. M. Sandborn, N. Quack, T. J. Seok, Y. Matsui, M. C. Wu, and B. E. Boser, “11.8 chip-scale electro-optical 3d fmcw lidar with 8 μ m ranging precision,” in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 214–216, Jan 2016.
- [49] C. Zhang, N. Hayashi, S. Y. Set, and S. Yamashita, “Polarization-insensitive amplitude-modulated cw lidar,” in *2019 Conference on Lasers and Electro-Optics (CLEO)*, pp. 1–2, May 2019.
- [50] J. P. Godbaz, M. J. Cree, A. A. Dorrington, and A. D. Payne, “A fast maximum likelihood method for improving amcw lidar precision using waveform shape,” in *SENSORS, 2009 IEEE*, pp. 735–738, Oct 2009.
- [51] P. Lindelow and J. J. Mohr, “Coherent lidar modulated with frequency stepped pulse trains for unambiguous high duty cycle range and velocity sensing in

BIBLIOGRAPHY

- the atmosphere,” in *2007 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2787–2790, July 2007.
- [52] GISGeography, “A complete guide to lidar: Light detection and rangingm.”
<https://gisgeography.com/lidar-light-detection-and-ranging/>.
[online] Accessed on 26/10/2019.
- [53] D. Li, M. Liu, R. Ma, and Z. Zhu, “An 8-ch lidar receiver based on tdc with multi-interval detection and real-time in-situ calibration,” *IEEE Transactions on Instrumentation and Measurement*, pp. 1–1, 2019.
- [54] S. Kurtti, J. Jansson, and J. Kostamovaara, “A cmos receiver—tdc chip set for accurate pulsed tof laser ranging,” *IEEE Transactions on Instrumentation and Measurement*, pp. 1–1, 2019.
- [55] R. sun, L. chen, and G. jin, “A gating system based on fpga for data acquisition of doppler wind lidar,” in *2019 IEEE 2nd International Conference on Electronics Technology (ICET)*, pp. 190–194, May 2019.
- [56] E. Arabul, A. Girach, J. Rarity, and N. Dahnoun, “Precise multi-channel timing analysis system for multi-stop lidar correlation,” in *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–6, Oct 2017.
- [57] SensL, “Sipm for automotive 3d imaging lidar systems.”
<https://www.sensl.com/downloads/ds/SensL-PW2018-POSTER.pdf>.
[online] Accessed on 10/11/2019.
- [58] J. Hecht, “Lasers for lidar: Fmcw lidar: An alternative for self-driving cars.”
<https://www.laserfocusworld.com/home/article/16556322/lasers-for-lidar-fmcw-lidar-an-alternative-for-selfdriving-cars>, May 2019.
[online] Accessed on 22/10/2019.
- [59] R. Hui and M. Osullivan, “Basic instrumentation for optical measurement,” *Fiber Optic Measurement Techniques*, p. 129–258, 2009.
- [60] N. W. University, “Quantum mechanics timeline.”
http://faculty.wcas.northwestern.edu/infocom/Ideas/quantum_timeline.html.
[online] Accessed on 23/09/2019.
- [61] J. D. Norton, “Origins of quantum theory.”
https://www.pitt.edu/jdnorton/teaching/HPS_0410/chapters/quantum_theory_origins/.
[online] Accessed on 23/09/2019.

- [62] T. E. o. E. Britannica, “Uncertainty principle.”
<https://www.britannica.com/science/uncertainty-principle>.
[online] Accessed on 23/09/2019.
- [63] A. Dawar, “Quantum computing lecture 1.”
<https://www.cl.cam.ac.uk/teaching/1213/QuantComp/lecture1.pdf>, Sep 2019.
[online] Accessed on 23/09/2019.
- [64] Y. Swami and AMRATA, “Recognition of quantum computers vs. classical computers,” 12 2011.
- [65] R. Fitzpatrick, “Quantum interference of light.”
<http://farside.ph.utexas.edu/teaching/qmech/Quantum/node22.html>.
[online] Accessed on 23/09/2019.
- [66] C. K. Hong, Z. Y. Ou, and L. Mandel, “Measurement of subpicosecond time intervals between two photons by interference,” *Phys. Rev. Lett.*, vol. 59, pp. 2044–2046, 1987.
- [67] A. M. Branczyk, “Hong-ou-mandel interference.”
<https://arxiv.org/pdf/1711.00080.pdf>.
[online] Accessed on 23/09/2019.
- [68] J.-L. Tambasco, G. Corrielli, R. J. Chapman, A. Crespi, O. Zilberberg, R. Osellame, and A. Peruzzo, “Quantum interference of topological states of light,” *Science Advances*, vol. 4, no. 9, 2018.
- [69] the Editors of Encyclopaedia Britannica, “Explainer: What is quantum communication?.” <https://www.technologyreview.com/s/612964/what-is-quantum-communications/>.
[online] Accessed on 3/12/2019.
- [70] M. Haitjema, “A survey of the prominent quantum key distribution protocols.”
<https://www.cse.wustl.edu/jain/cse571-07/ftp/quantum/>.
[online] Accessed on 3/12/2019.
- [71] A. K. Ekert, “Quantum cryptography based on bell’s theorem,” *Phys. Rev. Lett.*, vol. 67, pp. 661–663, Aug 1991.
- [72] M. A. Rubin and S. A. Kaushik, “Laser radar and quantum states of light,” *Coherent Optical Technologies and Applications*, 2008.
- [73] F. Grosshans and P. Grangier, “Continuous variable quantum cryptography using coherent states,” *Physical Review Letters*, vol. 88, no. 5, 2002.

BIBLIOGRAPHY

- [74] R. Paschotta, "Coherent states."
https://www.rp-photonics.com/coherent_states.html.
[online] Accessed on 3/12/2019.
- [75] M. Fujiwara, K. Tsujino, M. Akiba, and M. Sasaki, "Status of development of photon number resolving detectors," *Journal of the National Institute of Information and Communications Technology*, vol. 53, no. 1, 2006.
- [76] G. Wang, Z. Li, Y. Qiao, Z. Chen, X. Peng, and H. Guo, "Light source monitoring in quantum key distribution with single-photon detector at room temperature," *IEEE Journal of Quantum Electronics*, vol. 54, pp. 1–10, June 2018.
- [77] J. Hloušek, M. Dudka, I. Straka, and M. Ježek, "Accurate detection of arbitrary photon statistics," *Physical Review Letters*, vol. 123, Nov 2019.
- [78] R. Singh, "Fpga vs asic: Differences between them and which one to use?."
<https://numato.com/blog/differences-between-fpga-and-asics/>, Jul 2018.
[online] Accessed on 10/12/2019.
- [79] Xilinx, "What is an fpga?."
<https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>.
[online] Accessed on 23/09/2019.
- [80] Altera, "Integrated power solutions for altera fpgas."
<https://www.analog.com/media/en/news-marketing-collateral/solutions-bulletins-brochures/Integrated-Power-Solutions-for-Altera-FPGAs.pdf>.
[online] Accessed on 11/11/2019.
- [81] ElProCus, "Know about fpga architecture and thier applications."
<https://www.elprocus.com/fpga-architecture-and-applications/>, Oct 2014.
[online] Accessed on 23/09/2019.
- [82] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 203–215, Feb 2007.
- [83] HardwareBee, "List of fpga companies."
<http://hardwarebee.com/list-fpga-companies/>, Jun 2019.
[online] Accessed on 23/09/2019.
- [84] AnySilicon, "Fpga vs asic, what to choose?."
<https://any silicon.com/fpga-vs-asic-choose/>, Dec 2018.
[online] Accessed on 23/09/2019.

- [85] ElProCus, “Application specific integrated circuit: Types, and applications.”
<https://www.elprocus.com/application-specific-integrated-circuits/>, Jan 2016.
[online] Accessed on 23/09/2019.
- [86] J. Kalisz, “Review of methods for time interval measurements with picosecond resolution,” *Metrologia*, vol. 41, no. 1, p. 17, 2004.
- [87] D. Stoppa, F. Borghetti, J. Richardson, R. Walker, L. Grant, R. K. Henderson, M. Gersbach, and E. Charbon, “A 32x32-pixel array with in-pixel photon counting and arrival time measurement in the analog domain,” in *2009 Proceedings of ESSCIRC*, pp. 204–207, Sept 2009.
- [88] Velodyne, “How lidar technology enables autonomous cars to operate safely.”
<https://velodynelidar.com/newsroom/how-lidar-technology-enables-autonomous-cars-to-operate-safely/>, September 2018.
[online] Accessed on 23/09/2019.
- [89] R. Nock, N. Dahnoun, and J. Rarity, “Low cost timing interval analyzers for quantum key distribution,” in *2011 IEEE International Instrumentation and Measurement Technology Conference*, pp. 1–5, May 2011.
- [90] X. Hu, L. Zhao, S. Liu, J. Wang, and Q. An, “A stepped-up tree encoder for the 10-ps wave union tdc,” *IEEE Transactions on Nuclear Science*, vol. 60, pp. 3544–3549, Oct 2013.
- [91] A. Samarah and A. C. Carusone, “A digital phase-locked loop with calibrated coarse and stochastic fine tdc,” *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 1829–1841, Aug 2013.
- [92] D. Uchida, M. Ikebe, J. Motohisa, and E. Sano, “A 12-bit, 5.5- μ w single-slope adc using intermittent working tdc with multi-phase clock signals,” in *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 770–773, Dec 2014.
- [93] R. Nutts, “Digital time intervalometer with analogue vernier timing,”
US3541448A,1968-05-07.
- [94] S. Henzler, *Time-to-digital converters*, vol. 29.
Springer Science & Business Media, 2010.
- [95] A. Elkholy, T. Anand, W. S. Choi, A. Elshazly, and P. K. Hanumolu, “A 3.7 mw low-noise wide-bandwidth 4.5 ghz digital fractional-n pll using time amplifier-based tdc,” *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 867–881, April 2015.

BIBLIOGRAPHY

- [96] O. Sasaki, T. Taniguchi, T. K. Ohsaka, and H. Kurashige, "A high resolution tdc in tko box system," *IEEE Transactions on Nuclear Science*, vol. 35, pp. 342–347, Feb 1988.
- [97] B. K. Swann, B. J. Blalock, L. G. Clonts, D. M. Binkley, J. M. Rochelle, E. Breeding, and K. M. Baldwin, "A 100-ps time-resolution cmos time-to-digital converter for positron emission tomography imaging applications," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1839–1852, Nov 2004.
- [98] K. Maatta and J. Kostamovaara, "A high-precision time-to-digital converter for pulsed time-of-flight laser radar applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, pp. 521–536, Apr 1998.
- [99] J. Kostamovaara and R. Myllylä, "Time-to-digital converter with an analog interpolation circuit," *Review of Scientific Instruments*, vol. 57, no. 11, pp. 2880–2885, 1986.
- [100] P. Keranen, K. Maatta, and J. Kostamovaara, "Wide-range time-to-digital converter with 1-ps single-shot precision," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, pp. 3162–3172, Sept 2011.
- [101] Y. H. Seo, J. S. Kim, H. J. Park, and J. Y. Sim, "A 1.25 ps resolution 8b cyclic tdc in 0.13 μm cmos," *IEEE Journal of Solid-State Circuits*, vol. 47, pp. 736–743, March 2012.
- [102] S. Mandai and E. Charbon, "A 128-channel, 8.9-ps lsb, column-parallel two-stage tdc based on time difference amplification for time-resolved imaging," *IEEE Transactions on Nuclear Science*, vol. 59, pp. 2463–2470, Oct 2012.
- [103] K. Kim, Y. Kim, W. Yu, and S. Cho, "A 7b, 3.75ps resolution two-step time-to-digital converter in 65nm cmos using pulse-train time amplifier," in *2012 Symposium on VLSI Circuits (VLSIC)*, pp. 192–193, June 2012.
- [104] A. S. Yousif and J. W. Haslett, "A fine resolution tdc architecture for next generation pet imaging," *IEEE Transactions on Nuclear Science*, vol. 54, no. 5, pp. 1574–1582, 2007.
- [105] C. Favi and E. Charbon, "A 17ps time-to-digital converter implemented in 65nm fpga technology," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '09, (New York, NY, USA), pp. 113–120, ACM, 2009.

- [106] J. Wu, Z. Shi, and I. Wang, "Firmware-only implementation of time-to-digital converter (tdc) in field-programmable gate array (fpga)," *2003 IEEE Nuclear Science Symposium. Conference Record (IEEE Cat. No.03CH37515)*, 2003.
- [107] T. Sui, Z. Zhao, S. Xie, Y. Xie, Y. Zhao, Q. Huang, J. Xu, and Q. Peng, "A 2.3-ps rms resolution time-to-digital converter implemented in a low-cost cyclone v fpga," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, pp. 3647–3660, Oct 2019.
- [108] E. Nate, "Fpga – configurable logic block."
<https://blog.digilentinc.com/fpga-configurable-logic-block/>, 2020.
- [109] M. Mota and J. Christiansen, "A high-resolution time interpolator based on a delay locked loop and an rc delay line," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1360–1366, Oct 1999.
- [110] L. Perktold and J. Christiansen, "A multichannel time-to-digital converter asic with better than 3 ps rms time resolution," *Journal of Instrumentation*, vol. 9, no. 01, 2014.
- [111] T. Hashimoto, H. Yamazaki, A. Muramatsu, T. Sato, and A. Inoue, "Time-to-digital converter with vernier delay mismatch compensation for high resolution on-die clock jitter measurement," in *2008 IEEE Symposium on VLSI Circuits*, pp. 166–167, June 2008.
- [112] J. Yu, F. F. Dai, and R. C. Jaeger, "A 12-bit vernier ring time-to-digital converter in 0.13 μm cmos technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, p. 830–842, 2010.
- [113] M.-C. Lin, G.-R. Tsai, C.-Y. Liu, and S.-S. Chu, "Fpga-based high area efficient time-to-digital ip design," *TENCON 2006 - 2006 IEEE Region 10 Conference*, 2006.
- [114] M.-A. Daigneault and J. P. David, "Towards 5ps resolution tdc on a dynamically reconfigurable fpga (abstract only)," *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays - FPGA 10*, 2010.
- [115] K. Karadamoglou, N. P. Paschalidis, E. Sarris, N. Stamatopoulos, G. Kottaras, and V. Paschalidis, "An 11-bit high-resolution and adjustable-range cmos time-to-digital converter for space science instruments," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 214–222, Jan 2004.

BIBLIOGRAPHY

- [116] P. Chen and S.-I. Liu, "A cyclic cmos time-to-digital converter with deep sub-nanosecond resolution," in *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference (Cat. No.99CH36327)*, pp. 605–608, 1999.
- [117] S. Tisa, A. Lotito, A. Giudice, and F. Zappa, "Monolithic time-to-digital converter with 20ps resolution," in *ESSCIRC 2004 - 29th European Solid-State Circuits Conference (IEEE Cat. No.03EX705)*, pp. 465–468, Sept 2003.
- [118] Y. Liu, U. Vollenbruch, Y. Chen, C. Wicpalek, L. Maurer, T. Mayer, Z. Boos, and R. Weigel, "A 6ps resolution pulse shrinking time-to-digital converter as phase detector in multi-mode transceiver," in *2008 IEEE Radio and Wireless Symposium*, pp. 163–166, Jan 2008.
- [119] J. Zhang and D. Zhou, "An 8.5-ps two-stage vernier delay-line loop shrinking time-to-digital converter in 130-nm flash fpga," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, pp. 406–414, Feb 2018.
- [120] S. Pellerano, P. Madoglio, and Y. Palaskas, "A 4.75-ghz fractional frequency divider-by-1.25 with tdc-based all-digital spur calibration in 45-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 3422–3433, Dec 2009.
- [121] S. Henzler, S. Koeppe, D. Lorenz, W. Kamp, R. Kuenemund, and D. Schmitt-Landsiedel, "A local passive time interpolation concept for variation-tolerant high-resolution time-to-digital conversion," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 1666–1676, July 2008.
- [122] S. Henzler, S. Koeppe, D. Lorenz, W. Kamp, R. Kuenemund, and D. Schmitt-Landsiedel, "Variation tolerant high resolution and low latency time-to-digital converter," in *ESSCIRC 2007 - 33rd European Solid-State Circuits Conference*, pp. 194–197, Sept 2007.
- [123] M. S. Kim, Y. B. Kim, and K. K. Kim, "All-digital phased-locked loop with local passive interpolation time-to-digital converter based on a tristate inverter," in *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 326–329, Aug 2012.
- [124] S. Ito, S. Nishimura, H. Kobayashi, S. Uemori, Y. Tan, N. Takai, T. J. Yamaguchi, and K. Niitsu, "Stochastic tdc architecture with self-calibration," in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1027–1030, Dec 2010.
- [125] D. Kościelnik, M. Miśkowicz, J. Szyduczyński, and D. Rzepka, "Optimizing time-to-digital converter architecture for successive approximation time measurements," in *2013 IEEE Nordic-Mediterranean Workshop on Time-to-Digital Converters (NoMe TDC)*, pp. 1–8, Oct 2013.

- [126] D. Kościelnik, J. Szyduczyński, D. Rzepka, W. Andrysiewicz, and M. Miśkowicz, "Optimized design of successive approximation time-to-digital converter with single set of delay lines," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–8, June 2016.
- [127] A. Mantyniemi, T. Rahkonen, and J. Kostamovaara, "A cmos time-to-digital converter (tdc) based on a cyclic time domain successive approximation interpolation method," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 11, pp. 3067–3078, 2009.
- [128] A. Mäntyniemi and J. Kostamovaara, "Time-to-digital converter (tdc) based on startable ring oscillators and successive approximation," in *NORCHIP, 2014*, pp. 1–4, IEEE, 2014.
- [129] M. Z. Straayer and M. H. Perrott, "A multi-path gated ring oscillator tdc with first-order noise shaping," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 1089–1098, April 2009.
- [130] Y. Cao, P. Leroux, W. D. Cock, and M. Steyaert, "A 1.7mw 11b 1-1-1 mash $\delta\sigma$ time-to-digital converter," in *2011 IEEE International Solid-State Circuits Conference*, pp. 480–482, Feb 2011.
- [131] A. Elshazly, S. Rao, B. Young, and P. K. Hanumolu, "A 13b 315fsrms 2mw 500ms/s 1mhz bandwidth highly digital time-to-digital converter using switched ring oscillators," in *2012 IEEE International Solid-State Circuits Conference*, pp. 464–466, Feb 2012.
- [132] C. M. Hsu, M. Z. Straayer, and M. H. Perrott, "A low-noise wide-bw 3.6-ghz digital *deltastigma* fractional-n frequency synthesizer with a noise-shaping time-to-digital converter and quantization noise cancellation," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 2776–2786, Dec 2008.
- [133] M. Z. Straayer and M. H. Perrott, "An efficient high-resolution 11-bit noise-shaping multipath gated ring oscillator tdc," in *2008 IEEE Symposium on VLSI Circuits*, pp. 82–83, June 2008.
- [134] P. Lu and P. Andreani, "A high-resolution vernier gated-ring-oscillator tdc in 90-nm cmos," in *NORCHIP 2010*, pp. 1–4, Nov 2010.
- [135] P. Keränen and J. Kostamovaara, "Algorithmic time-to-digital converter," in *2013 NORCHIP*, pp. 1–4, Nov 2013.
- [136] P. Keränen and J. Kostamovaara, "A wide range, 4.2 ps(rms) precision cmos tdc with cyclic interpolators based on switched-frequency ring oscillators," *IEEE*

BIBLIOGRAPHY

- Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 2795–2805, Dec 2015.
- [137] M. Bogdan, H. Frisch, M. Heintz, A. Paramonov, H. Sanders, S. Chappa, R. DeMaat, R. Klein, T. Miao, P. Wilson, and T. J. Phillips, “A 96-channel fpga-based time-to-digital converter (tdc) and fast trigger processor module with multi-hit capability and pipeline,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 554, no. 1, pp. 444 – 457, 2005.
- [138] I. Konorov, “iftdc architecture,” *COMPASS DAQFEET 2019*, vol. 43, no. 4, pp. 855–863, 2019.
- [139] J. Wu and Z. Shi, “The 10-ps wave union tdc: Improving fpga tdc resolution beyond its cell delay,” in *2008 IEEE Nuclear Science Symposium Conference Record*, pp. 3440–3446, Oct 2008.
- [140] S. Tancock, E. Arabul, N. Dahnoun, and S. Mehmood, “Can dsp48a1 adders be used for high-resolution delay generation?,” in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–6, June 2018.
- [141] S. Tancock and N. Dahnoun, “A 5.25ps-resolution tdc on fpga using dsp blocks,” in *Proceedings of the Digital Image & Signal Processing '19 conference*, Springer, 10 2018.
- [142] T. BMJ, “Mean and standard deviation.”
<https://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/2-mean-and-standard-deviation>,
2019.
[online] Accessed on 23/09/2019.
- [143] P. Harpe, *High-performance ad and da converters, ic design in scaled technologies, and time-domain signal ... processing*.
Springer International Pu, 2016.
- [144] I. C. I. of Technology, “The gaussian or normal distribution.”
https://ned.ipac.caltech.edu/level5/Leo/Stats2_3.html.
[online] Accessed on 23/09/2019.
- [145] S. Arar, “What are the dnl and inl specifications of a dac? non-linearity in digital-to-analog converters.”
<https://www.allaboutcircuits.com/technical-articles/understanding-dnl-and-inl-specifications-of-a-digital-to-analog-converter/>, Mar
2019.
[online] Accessed on 23/09/2019.

- [146] L. Zaworski, D. Chaberski, M. Kowalski, and M. Zielinski, "Quantization error in time-to-digital converters," *Metrology and Measurement Systems*, vol. 19, no. 1, 2012.
- [147] J. Tangudu, S. Gunturi, S. Jalan, J. Janardhanan, R. Ganesan, D. Sahu, K. Waheed, J. Wallberg, and R. B. Staszewski, "Quantization noise improvement of time to digital converter (tdc) for adpll," in *2009 IEEE International Symposium on Circuits and Systems*, pp. 1020–1023, May 2009.
- [148] D. K. Tala, "What is metastability?."
<http://www.asic-world.com/tidbits/metastablity.html>, 2014.
[online] Accessed on 15/10/2019.
- [149] M. Függer, A. Kinali, C. Lenzen, and T. Polzer, "Metastability-aware memory-efficient time-to-digital converters," in *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 49–56, May 2017.
- [150] S. Beer and R. Ginosar, "Eleven ways to boost your synchronizer," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, p. 1040–1049, 2015.
- [151] S. Friedrichs, M. Fugger, and C. Lenzen, "Metastability-containing circuits," *IEEE Transactions on Computers*, vol. 67, p. 1167–1183, Jan 2018.
- [152] G. Tarawneh and A. Yakovlev, "An rtl method for hiding clock domain crossing latency," *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*, 2012.
- [153] C. Lenzen and M. Medina, "Efficient metastability-containing gray code 2-sort," *2016 22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2016.
- [154] K. Katoh and K. Namba, "A low area calibration technique of tdc using variable clock generator for accurate on-line delay measurement," in *Sixteenth International Symposium on Quality Electronic Design*, pp. 430–434, March 2015.
- [155] R. W. R. Nock, *Flexible precision timing instrumentation and Quantum Key Distribution*.
Phd thesis, 2013.
- [156] M. W. Fishburn and E. Charbon, "Statistical limitations of tdc density tests," in *In Proc. IEEE Nuclear Science Symposium (NSS)*, October 2012.

BIBLIOGRAPHY

- [157] M. Mota and J. Christiansen, "A four-channel self-calibrating high-resolution time to digital converter," in *1998 IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology (Cat. No.98EX196)*, vol. 1, pp. 409–412 vol.1, 1998.
- [158] R. Szplet, Z. Jachna, P. Kwiatkowski, and K. Rozyc, "A 2.9 ps equivalent resolution interpolating time counter based on multiple independent coding lines," *Measurement Science and Technology*, vol. 24, no. 3, p. 035904, 2013.
- [159] Q. Shen, S. Liu, B. Qi, Q. An, S. Liao, P. Shang, C. Peng, and W. Liu, "A 1.7 ps equivalent bin size and 4.2 ps rms fpga tdc based on multichain measurements averaging method," *IEEE Transactions on Nuclear Science*, vol. 62, pp. 947–954, June 2015.
- [160] B. Markovic, S. Tisa, F. A. Villa, A. Tosi, and F. Zappa, "A high-linearity, 17 ps precision time-to-digital converter based on a single-stage vernier delay loop fine interpolation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, pp. 557–569, March 2013.
- [161] S. Alahdab, A. Mäntyniemi, and J. Kostamovaara, "A time-to-digital converter (tdc) with a 13-bit cyclic time domain successive approximation interpolator with sub-ps-level resolution using current dac and differential switch," in *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 828–831, Aug 2013.
- [162] V. Kratyuk, P. K. Hanumolu, K. Ok, U. K. Moon, and K. Mayaram, "A digital pll with a stochastic time-to-digital converter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, pp. 1612–1621, Aug 2009.
- [163] J. S. Teh, L. Siek, A. M. Alonso, A. Firdauzi, and A. Matsuzawa, "A 14-b, 850fs fully synthesizable stochastic-based branching time-to-digital converter in 65nm cmos," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2018.
- [164] M. Lee and A. A. Abidi, "A 9b, 1.25ps resolution coarse-fine time-to-digital converter in 90nm cmos that amplifies a time residue," in *2007 IEEE Symposium on VLSI Circuits*, pp. 168–169, June 2007.
- [165] E. Bayer and M. Traxler, "A high-resolution (< 10 ps rms) 48-channel time-to-digital converter (tdc) implemented in a field programmable gate array (fpga)," *IEEE Transactions on Nuclear Science*, vol. 58, pp. 1547–1552, Aug 2011.

- [166] V. Ramakrishnan and P. T. Balsara, "A wide-range, high-resolution, compact, cmos time to digital converter," in *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*, pp. 6 pp.–, Jan 2006.
- [167] M. Safi-Harb and G. W. Roberts, "Embedded narrow pulse measurement in digital cmos," in *2006 IEEE Instrumentation and Measurement Technology Conference Proceedings*, pp. 1195–1200, 2006.
- [168] A. Mantyniemi, T. Rahkonen, and J. Kostamovaara, "A cmos time-to-digital converter (tdc) based on a cyclic time domain successive approximation interpolation method," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 3067–3078, Nov 2009.
- [169] M. Takayama, S. Dosho, N. Takeda, M. Miyahara, and A. Matsuzawa, "A time-domain architecture and design method of high speed a-to-d converters with standard cells," *IEEE Asian Solid-State Circuits Conference 2011*, 2011.
- [170] R. B. Staszewski, S. Vemulapalli, P. Vallur, J. Wallberg, and P. T. Balsara, "1.3 v 20 ps time-to-digital converter for frequency synthesis in 90-nm cmos," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 3, pp. 220–224, 2006.
- [171] C.-S. Hwang, P. Chen, and H.-W. Tsao, "A high-precision time-to-digital converter using a two-level conversion scheme," *IEEE Transactions on Nuclear Science*, vol. 51, pp. 1349–1352, Aug 2004.
- [172] P. Dudek, S. Szczepanski, and J. V. Hatfield, "A high-resolution cmos time-to-digital converter utilizing a vernier delay line," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 240–247, Feb 2000.
- [173] M. Rashdan, A. Yousif, J. Haslett, and B. Maundy, "A new time-based architecture for serial communication links," in *2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)*, pp. 531–534, Dec 2009.
- [174] E. Raisanen-Ruotsalainen, T. Rahkonen, and J. Kostamovaara, "An integrated time-to-digital converter with 30-ps single-shot precision," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1507–1510, Oct 2000.
- [175] R. B. Staszewski, K. Muhammad, D. Leipold, C.-M. Hung, Y.-C. Ho, J. L. Wallberg, C. Fernando, K. Maggio, R. Staszewski, T. Jung, J. Koh, S. John, I. Y. Deng, V. Sarda, O. Moreira-Tamayo, V. Mayega, R. Katz, O. Friedman, O. E. Eliezer, E. de Obaldia, and P. T. Balsara, "All-digital tx frequency

BIBLIOGRAPHY

- synthesizer and discrete-time receiver for bluetooth radio in 130-nm cmos,” *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 2278–2291, Dec 2004.
- [176] B. M. Helal, M. Z. Straayer, G. Wei, and M. H. Perrott, “A highly digital mdll-based clock multiplier that leverages a self-scrambling time-to-digital converter to achieve subpicosecond jitter performance,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 855–863, 2008.
- [177] J. Richardson, R. Walker, L. Grant, D. Stoppa, F. Borghetti, E. Charbon, M. Gersbach, and R. K. Henderson, “A 32x32 50ps resolution 10 bit time to digital converter array in 130nm cmos for time correlated imaging,” in *2009 IEEE Custom Integrated Circuits Conference*, pp. 77–80, Sept 2009.
- [178] C. Veerappan, J. Richardson, R. Walker, D. Li, M. W. Fishburn, Y. Maruyama, D. Stoppa, F. Borghetti, M. Gersbach, R. K. Henderson, and E. Charbon, “A 160x128 single-photon image sensor with on-pixel 55ps 10b time-to-digital converter,” in *2011 IEEE International Solid-State Circuits Conference*, pp. 312–314, 2011.
- [179] I. Nissinen and J. Kostamovaara, “Time-to-digital converter based on an on-chip voltage reference locked ring oscillator,” in *2006 IEEE Instrumentation and Measurement Technology Conference Proceedings*, pp. 250–254, April 2006.
- [180] D. K. Xie, Q. C. Zhang, G. S. Qi, and D. Y. Xu, “Cascading delay line time-to-digital converter with 75 ps resolution and a reduced number of delay cells,” *Review of Scientific Instruments*, vol. 76, no. 1, p. 014701, 2005.
- [181] V. Dhanasekaran, M. Gambhir, M. M. Elsayed, E. Sanchez-Sinencio, J. Silva-Martinez, C. Mishra, L. Chen, and E. Pankratz, “A 20mhz bw 68db dr ct $\delta\sigma$ adc based on a multi-bit time-domain quantizer and feedback element,” in *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pp. 174–175, 175a, Feb 2009.
- [182] C. Niclass, C. Favi, T. Kluter, M. Gersbach, and E. Charbon, “A 128 times 128 single-photon image sensor with column-level 10-bit time-to-digital converter array,” *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 2977–2989, Dec 2008.
- [183] D. M. Santos, S. F. Dow, J. M. Flasek, and M. E. Levi, “A cmos delay locked loop and sub-nanosecond time-to-digital converter chip,” *IEEE Transactions on Nuclear Science*, vol. 43, pp. 1717–1719, Jun 1996.
- [184] C. P. University, “Introduction to geiger counters.”
<https://www.cpp.edu/~pbsiegel/bio431/texnotes/chapter4.pdf>
[online] Accessed on 23/09/2019.

- [185] W. Bothe, “The coincidence method.”
<https://www.nobelprize.org/prizes/physics/1954/bothe/lecture/>, 1954.
[online] Accessed on 21/10/2019.
- [186] A. D. Aliaga Kelly, W. Boyes, “Coincidence circuits.”
<https://www.sciencedirect.com/topics/physics-and-astronomy/coincidence-circuits>,
2010.
[online] Accessed on 21/10/2019.
- [187] Vysochanskij, Mukhin, Tsun, Tsun, Semenyushkin, and I.n., “A multi-channel coincidence circuit with short time intervals,” *International Atomic Energy Agency (IAEA)*, Jan 1962.
- [188] K. Taguchi, M. Zhang, E. C. Frey, X. Wang, J. S. Iwanczyk, E. Nygard, N. E. Hartsough, B. M. W. Tsui, and W. C. Barber, “Modeling the performance of a photon counting x-ray detector for ct: Energy response and pulse pileup effects,” *Medical Physics*, vol. 38, p. 1089–1102, Jan 2011.
- [189] S. Hsieh and H. Chou, “A digital coincidence measurement system using fpga techniques,” *2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, 2009.
- [190] “414a fast coincidence: Products: Ametek ortec.” <https://www.ortec-online.com/products/electronics/delays-gates-and-logic-modules/414a>.
[online] Accessed on 26/09/2019.
- [191] M. F. Masters, T. Heral, and K. Tummala, “Low-cost coincidence counting apparatus for single photon optics investigations,” *2015 Conference on Laboratory Instruction Beyond the First Year*, 2015.
- [192] T. Kim, M. Fiorentino, P. V. Gorelik, and F. N. C. Wong, “Low-cost nanosecond electronic coincidence detector,” 2005.
- [193] D. Branning, S. Bhandari, and M. Beck, “Low-cost coincidence-counting electronics for undergraduate quantum optics,” *American Journal of Physics - AMER J PHYS*, vol. 77, pp. 667–670, 01 2009.
- [194] S. Gaertner, H. Weinfurter, and C. Kurtsiefer, “Fast and compact multichannel photon coincidence unit for quantum information processing,” *Review of Scientific Instruments*, vol. 76, no. 12, p. 123108, 2005.
- [195] L. Bai and W. Zhou, “The measurement of transient stability with high resolution,” in *2013 Joint European Frequency and Time Forum*

BIBLIOGRAPHY

- International Frequency Control Symposium (EFTF/IFC)*, pp. 175–178, July 2013.
- [196] P. Lecoq, “Pushing the limits in time-of-flight pet imaging,” *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 1, no. 6, p. 473–485, 2017.
- [197] M. Ware, A. Migdall, J. C. Bienfang, and S. V. Polyakov, “Calibrating photon-counting detectors to high accuracy: background and deadtime issues,” *Journal of Modern Optics*, vol. 54, no. 2-3, p. 361–372, 2007.
- [198] Xilinx, “Ultrascale fpga, product tables and product selection guide.”
<https://www.xilinx.com/support/documentation/selection-guides/ultrascale-fpga-product-selection-guide.pdf>,
2016.
[online] Accessed on 23/09/2019.
- [199] S. Miki, S. Miyajima, M. Yabuno, T. Yamashita, T. Yamamoto, N. Imoto, R. Ikuta, R. A. Kirkwood, R. H. Hadfield, H. Terai, and et al., “Timing jitter characterization of the sfq coincidence circuit by optically time-controlled signals from sspds,” *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 5, p. 1–4, 2019.
- [200] S. Miyajima, S. Miki, M. Yabuno, T. Yamashita, and H. Terai, “Timing discriminator based on single-flux-quantum circuit toward high time-resolved photon detection,” *Superconductor Science and Technology*, vol. 30, no. 12, 2017.
- [201] M.-J. Lee, W. Dally, T. Greer, H.-T. Ng, R. Farjad-Rad, J. Poulton, and R. Senthinathan, “Jitter transfer characteristics of delay-locked loops - theories and design techniques,” *IEEE Journal of Solid-State Circuits*, vol. 38, no. 4, p. 614–621, 2003.
- [202] J. Wu and Z. Shi, “The 10-ps wave union tdc: Improving fpga tdc resolution beyond its cell delay,” *2008 IEEE Nuclear Science Symposium Conference Record*, 2008.
- [203] Hossein Pishro-Nik, “4.1.1 Probability Density Function (PDF).”
https://www.probabilitycourse.com/chapter4/4_1_1_pdf.php.
[online] Accessed on 23/09/2019.
- [204] T. Keary, “1.2: Data rate, throughput and bandwidth.”
<https://www.comparitech.com/net-admin/throughput-vs-bandwidth/>.
[online] Accessed on 10/08/2020.

- [205] IQT , “PCI Express™ Jitter Attenuator.”
<https://www.idt.com/document/dst/874001i-02-datasheet>, 2010.
[online] Accessed on 23/09/2019.
- [206] IDT , “VCTCXO Specification IQXT-210.”
<https://www.idt.com/document/dst/874001i-02-datasheet>, 2010.
[online] Accessed on 23/09/2019.
- [207] OpalKelly, “Frontpanel api.”
<https://docs.opalkelly.com/display/FPSDK/FrontPanel+API>.
[online] Accessed on 10/11/2019.
- [208] J. W. Silverstone, D. Bonneau, K. Ohira, N. Suzuki, H. Yoshida, N. Iizuka, M. Ezaki, C. M. Natarajan, M. G. Tanner, R. H. Hadfield, *et al.*, “On-chip quantum interference between silicon photon-pair sources,” *Nature Photonics*, vol. 8, no. 2, p. 104, 2014.
- [209] A. E. Ulanov, I. A. Fedorov, D. Sychev, P. Grangier, and A. Lvovsky, “Loss-tolerant state engineering for quantum-enhanced metrology via the reverse hong–ou–mandel effect,” *Nature Communications*, vol. 7, p. 11925, 2016.
- [210] D. Bonneau, J. W. Silverstone, and M. G. Thompson, *Silicon Quantum Photonics*.
Springer, 2016.
- [211] W. H. Pernice, C. Schuck, O. Minaeva, M. Li, G. Goltsman, A. Sergienko, and H. Tang, “High-speed and high-efficiency travelling wave single-photon detectors embedded in nanophotonic circuits,” *Nature Communications*, vol. 3, p. 1325, 2012.

Appendix A : Results from Legacy Coincidence Counters

Introduction

This Appendix compiles results obtained from obsolete coincidence counting schemes. These schemes include the real-time coincidence counting software, the FPGA based backwards looking tag difference coincidence counter and the FIFO based forward looking tag difference coincidence counter.

Real-Time Coincidence Counting Software

Coincidence Counting Benchmarks

Results obtained from the coincidence counter software [12] will be compiled in this section. To test this system, the coincidence rate as a function of delay graph was plotted using both rolling window and forward looking tag difference methods. For testing, 512 kHz signal was generated to feed two channels with it and the integration time was around 1 s. The delay range was between -4ns and +4ns. For the results obtained from the rolling window approach can be seen in Figure 1 [12]. For tests conducted with the forward looking tag difference method can be found in Figure 2.

The smallest coincidence window size which could capture all the coincidences were measured as 0.49 ns with the rolling window. With the forward looking tag difference method, 0.12 ns was achieved, which proved it as a plausible method for the coincidence counting [12].

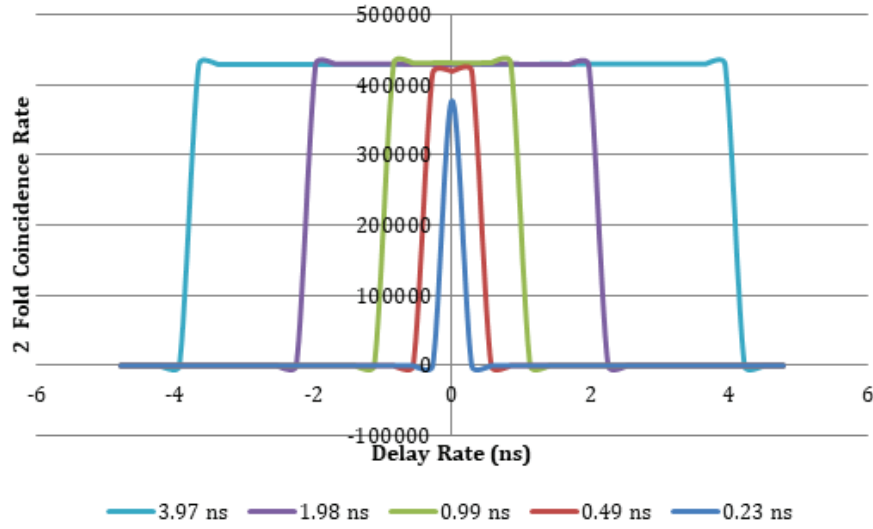


Figure 1: *The coincidence rate as a function of delay with Rolling Window [12]*

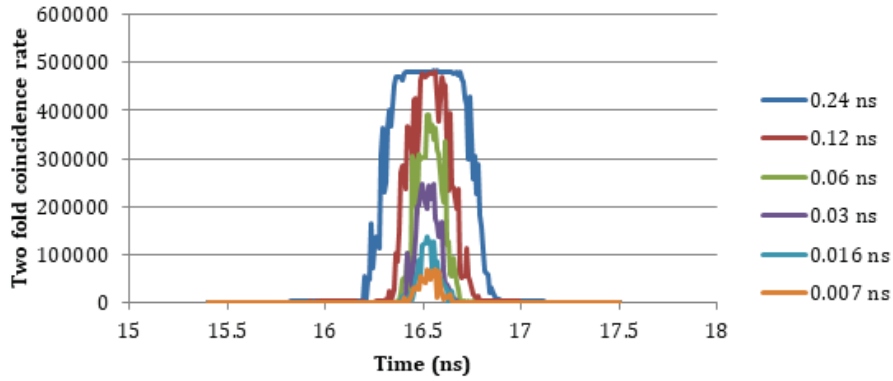


Figure 2: *The coincidence rate as function of delay with Forward Looking Tag Difference method [12]*

Backward Looking Tag Difference Real-Time FPGA based Coincidence Counter

Coincidence Counting Benchmarks

In this section, results obtained from the coincidence rate as a function of delay test for the backward looking tag difference method was compiled. This test was conducted by using a signal generator running at around 1 MHz, and a fixed delay was between two channels. The coincidence counter searched inside of a RAM block to find coincidences. The Integration time was 100 ms. Results for different window sizes could be seen in Figure 3 and in Figure 4 [11].

BACKWARD LOOKING TAG DIFFERENCE REAL-TIME FPGA BASED COINCIDENCE COUNTER

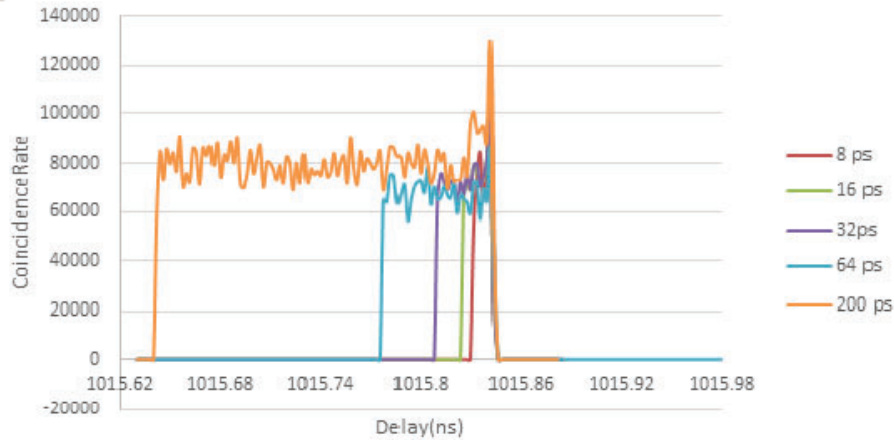


Figure 3: *The coincidence rate backward looking tag difference method [11]*

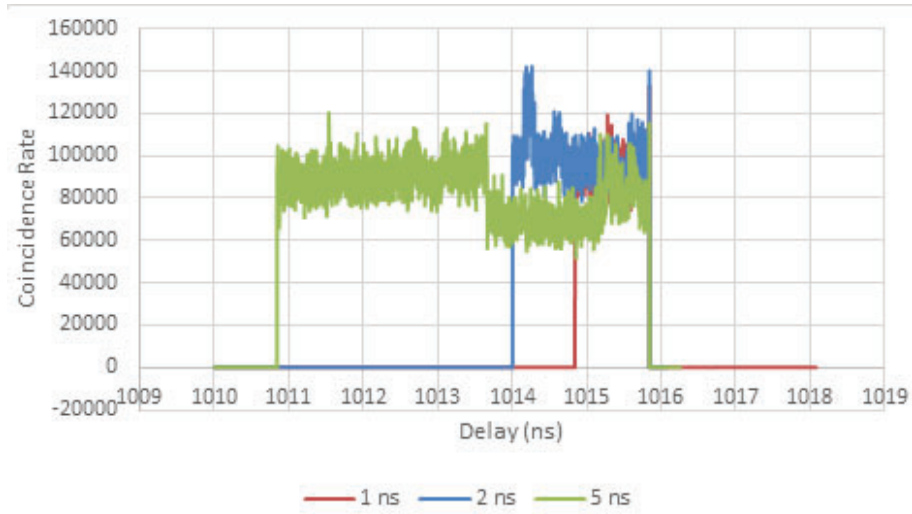


Figure 4: *The coincidence rate backward looking tag difference method [11]*

A very high jitter was observed at coincidence rates with this method since it has shown that iterating inside a RAM block for a coincidence detection is not ideal for the coincidence counting operation due to problems such as collisions of memory pointers and the slow data processing.

Real-Time Forward Looking Tag Difference FPGA using Tag Serialisation

Coincidence Counting Benchmarks

The final multi-channel coincidence counting method was initially implemented by using a single coincidence detecting and counting module where a FIFO was used for serialising tags as it was mentioned before. This implementation was also tried with the coincidence rate as a function of delay. For this test, two in-sync signal generators with 1.13MHz inputs were used, where the delay between them was set around 75 ns. The integration time was set to 600 ms, and around 680,000 tags were generated for both channels. By adding 15 ps delay every 600 ms and the following plots were achieved for this implementation [7].

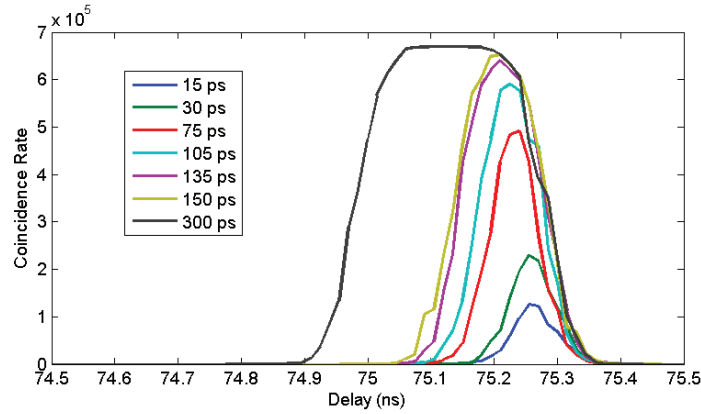


Figure 5: *The coincidence rate as a function of the delay [7]*

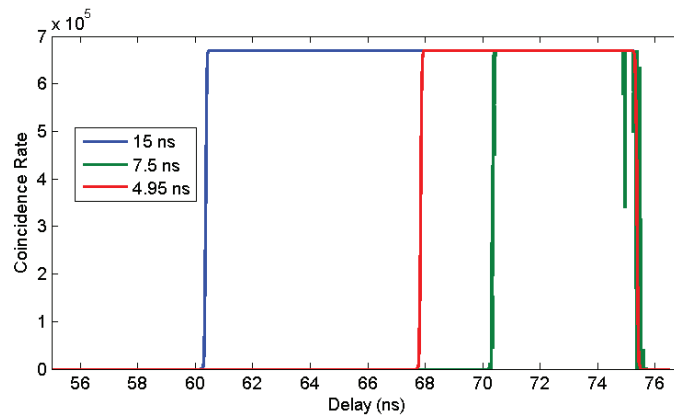


Figure 6: *The coincidence rate as a function of the delay [7]*

In Figure 5, the smallest window size which managed to capture all generated

REAL-TIME FORWARD LOOKING TAG DIFFERENCE FPGA USING TAG SERIALISATION

coincidences accurately was 150 ps in this scheme. With this scheme, the jitter problem was solved mentioned in section 8.3.3. Also, the performance of this scheme with larger coincidence window sizes can be seen in Figure 6.

Appendix B : Experiments Results From Pseudo-photon-number Resolving Detection of a Coherent State of Light

In this appendix, experiment results obtained from the pseudo-photon-number resolving detection of a coherent state of lights. Attenuation values are additional attenuation values added with VOA. For the results Table 1 can be referred.

**APPENDIX B : EXPERIMENTS RESULTS FROM
PSEUDO-PHOTON-NUMBER RESOLVING DETECTION OF A
COHERENT STATE OF LIGHT**

Attenuation	1-fold	2-fold	3-fold	4-fold	5-fold	6-fold	7-fold	8-fold
24 dB	42572700.45	2464893.90	566364.58	74217.47	5082.92	156.70	1.224	0
22 dB	45726658.27	3522706.87	1089620.65	153392.72	10022.17	270.54	0.723667	0
21 dB	56090323.4	5454398.48	2187740.96	384374.89	31648.01	1082.82	4.17	0.0556
20 dB	52778099.67	4703416.70	1546901.16	192134.67	7656.39	39.46	0.055	0
19.2 dB	45912733.85	3677679.74	959861.79	82511.07	256.73	1.28	0	0
18.5 dB	48453960.33	4176423.67	1121866.98	98708.35	309.61	1.55	0	0

Table 1: *Measurement results from Pseudo-photon-number Resolving Detection of a Coherent State of Light (Attenuation vs N-fold Coincidence Rate / s)*